Universidad
Carlos III de Madrid

Departamento de Ingeniería Telemática

PROYECTO FIN DE CARRERA

# DESARROLLO DE UNA INTERFAZ GRÁFICA DE USUARIO PARA HTML2XHTML

Autor: Juan Carlos González Haro

Tutor: Jesús Arias Fisteus

Leganés, Julio de 2010

**DESARROLLO DE UNA GUI  PARA HTML2XHTML**

Título: **DESARROLLO DE UNA INTERFAZ GRÁFICA DE USUARIO PARA HTML2XHTML**

Autor: Juan Carlos González Haro
Director: Jesús Arias Fisteus

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

**DESARROLLO DE UNA GUI  PARA HTML2XHTML**

A Javier por su aguante y porque sin su empeño no hubiera terminado este proyecto

A mi hermana Gema por su ayuda y experiencia

A mis padres por permitirme llegar a ser lo que soy

A mi familia y amigos

Desarrollo de una interfaz gráfica de usuario para html2xhtml es un proyecto fin de carrera realizado por Juan Carlos González Haro cuyo director es Jesús Arias Fisteus.

En estas páginas se abordará el problema de la generación de código en C# para Linux y sus posibles soluciones así como se implementará una pequeña aplicación para mostrar su uso.

Se realizará una interfaz gráfica para la aplicación html2xhtml en lenguaje C# que sea compatible tanto para sistemas Windows como Linux. Para ello, se va a utilizar el proyecto Mono. Este proyecto provee de bibliotecas para la ejecución de programas escritos en C#, con el entorno .NET de Microsoft, para que puedan ser ejecutados en entornos Linux. Además se realizará un repaso a las tecnologías existentes para este cometido y se investigará a fondo el Proyecto Mono de Software Libre que lo permite.

# INDICE

# INTRODUCCIÓN

## MOTIVACIÓN

Para poder publicar la información en Internet, se necesita un lenguaje que las máquinas entiendan. Este lenguaje se llama **HTML** (HyperText Markup Language).

**XHTML** es uno de los lenguajes derivados de XML usados en las páginas Web de Internet (o Intranets). XHTML está basado en HTML, pero con la diferencia que XHTML es más estricto y de acuerdo con el estándar XML. De hecho, la primera versión del XHTML es una mera adaptación del HTML a la sintaxis del XML. XHTML sigue muchos de los convenios de HTML, por ende los navegadores que soporten HTML podrán ver los documentos XHTML. Sin embargo, algunos navegadores más antiguos no pueden presentar XHTML con CSS apropiadamente.

HTML y XHTML permiten:

- Publicar documentos.
- Acceder a documentos cargándolos directamente en el navegador o, simplemente, activando en otros documentos hiperenlaces que apunten a él.
- Rellenar formularios con información que es enviada al servidor, para que este la procese y realice cierta tarea con ella.
- Incluir objetos multimedia (vídeos, imágenes, sonidos, animaciones, etc.) en los documentos.

Hasta la llegada de HTML 5, el W3C (World Wide Web Consortium), el principal organismo de estandarización en el ámbito de la Web, consideraba XHTML como la última versión de HTML y recomienda reemplazar el uso del HTML por XHTML, ya soportado por todos los navegadores.

El lenguaje XHTML es casi idéntico al HTML versión 4.01, por lo que la transición no es compleja, aunque sí costosa si tenemos que convertir cientos de páginas manualmente.

Existen algunas aplicaciones que automatizan la conversión de HTML en XHTML una de ellas es HTML2XHTML.

**HTML2XHTML** es una aplicación realizada por el profesor Jesús Arias Fisteus y su equipo dentro del **Departamento de Ingeniería Telemática** de la **Universidad Carlos III de Madrid.** Este programa traduce automáticamente documentos en formato HTML, a su correspondiente XHTML. Además incorpora una serie de opciones para que sea el fichero que resulta sea utilizado por el cliente apropiado (navegador Web en equipo de escritorio, móvil, impresora, etc.).

Esta pequeña aplicación, está construida para ejecutarse en consola, ya sea en Windows o Linux (el mismo código fuente compilado en distintos entornos), y puede resultar algo engorroso el ejecutar el programa con todas las opciones disponibles para ello, por lo que sería necesario una interfaz gráfica que posibilite un uso por un número mayor de usuarios.

# OBJETIVOS DEL PROYECTO

El **objetivo de este proyecto** es desarrollar una interfaz gráfica para html2xhtml y facilitar su uso por parte de los usuarios. De este modo, las ejecuciones de la aplicación en consola serán totalmente invisibles para el usuario. Además la interfaz gráfica deberá funcionar tanto en Windows como en Linux. Para ello, haremos uso del **Proyecto Mono**: unas bibliotecas de software libre para poder ejecutar aplicaciones escritas en C# utilizando una distribución Linux. Sobre el Proyecto Mono hablaremos más adelante. Por lo tanto, otro de los objetivos de este estudio, es evaluar el funcionamiento del Proyecto Mono sobre C# y comprobar, de primera mano, si es posible ejecutar una aplicación escrita en Código C# en Linux mediante el entorno de ejecución de Mono.

Además, en nuestra aplicación del motor gráfico, se le añadirán diversas mejoras para hacer más fácil y comprensible al usuario **HTML2XHTML** como que la interfaz gráfica debe facilitar tareas como la conversión de múltiples ficheros a la vez o la posibilidad de editar el resultado de la conversión.

# PLAN DE TRABAJO

Para llevar a cabo este plan de trabajo, se deben realizar distintas tareas como se apunta en el diagrama de Gannt adjunto (Figura 1).

La lista de tareas es la que se muestra a continuación:

- **Estudio del problema planteado**. En esta tarea se realizará el estudio previo de lo que se pide y se plantearán los objetivos y las siguientes tareas a realizar.
- **Estudio de Tecnologías Web (HTML, XHTML, VISUAL C#).** Esta parte del plan de trabajo se dirige a estudiar en profundidad las tecnologías necesarias para poder resolver el problema planteado en este proyecto.
- **Estudio de requerimientos.** Esta tarea se dedica a investigar lo requisitos necesarios para que el aplicación cumpla su cometido.
- **Realización de vistas de la Aplicación.** En esta tarea se desarrollarán todas las vistas del programa y su conexión entre ellas, si es que existen, para el buen funcionamiento de su entorno gráfico. Aquí estarían también realizar la función de colorear código.
- **Recoger variables de la aplicación.** En esta tarea se recogen los datos necesarios para formar la llamada a la aplicación html2xhtml.exe de forma correcta.
- **Realización de la ejecución de la cadena del programa.** En esta tarea se crea el código necesario para la llamada a *html2xhtml* y se envían y recogen los datos y resultados del programa para mostrarlos al usuario.
- **Desarrollo de la conversión múltiple de archivos.** En esta tarea se realiza el código necesario para que la aplicación tenga la funcionalidad de convertir múltiples archivos de forma automática.
- **Desarrollo de pruebas.** En esta parte del proyecto, sometemos a la aplicación a una serie de pruebas con distintos archivos para comprobar el buen funcionamiento de ésta.
- **Desarrollo de la memoria.** Durante todo el proceso de este proyecto, se realiza de forma simultánea esta memoria.

En cuanto al desarrollo de la aplicación en si, existen varias tareas que se describen a continuación.

Lo primero a realizar serían las vistas de la aplicación. Esto incluye una vista inicial con saludo, una vista o varias, según el caso, para cada el código fuente de los ficheros con las opciones obligatorias de traducción, una vista con las opciones no principales, y otra vista donde podremos traducir varios archivos de forma simultánea.

La siguiente tarea estaría encaminada a la conexión del sistema de ventanas con el archivo **HTMLXHTML**, de forma que pueda interactuar con la aplicación. Para esto se debería realizar las siguientes tareas:

- Recoger en variables las distintas opciones que tiene **HTMLXHTML**, para que logre funcionar.
- Realizar la cadena de ejecución del programa basado en consola (donde introduciremos las opciones recogidas así como el archivo a traducir).
- Realizar una redirección de la ejecución de tal manera que no guarde el resultado de la traducción en otro fichero sino que lo vuelva a sacar a vista.
- Realizar la parte de la traducción simultánea de los archivos.
- Realizar cambios en la parte de ventanas como son: añadir coloreado de código fuente, añadir menús contextuales…

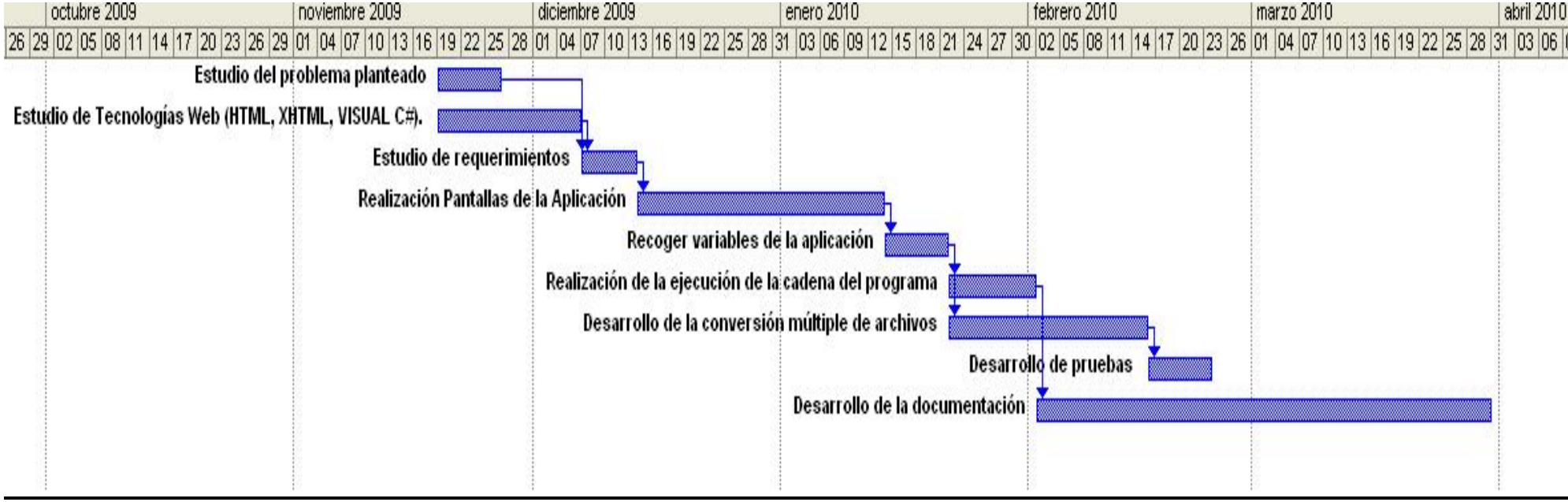



**Figura 1: Diagrama de Gantt del proyecto**

La duración de las tareas así como sus predecesores, queda como sigue:

| | TAREAS | DURACIÓN | COMIENZO | FIN | PREDECESORES |
|---|---|---|---|---|---|
| 1 | **Estudio del problema planteado** | 6d | jue 19/11/09 | vie 27/11/09 | |
| 2 | **Estudio de Tecnologías Web (HTML, XHTML, VISUAL C#).** | 12d | jue 19/11/09 | lun 07/12/09 | |
| 3 | **Estudio de requerimientos** | 5d | lun 07/12/09 | lun 14/12/09 | "1;2" |
| 4 | **Realización Vistas de la Aplicación** | 23d | lun 14/12/09 | jue 14/01/10 | 3 |
| 5 | **Recoger variables de la aplicación** | 6d | jue 14/01/10 | vie 22/01/10 | 4 |
| 6 | **Realización de la ejecución de la cadena del programa** | 7d | vie 22/01/10 | mar 02/02/10 | 5 |
| 7 | **Desarrollo de la conversión múltiple de archivos** | 17d | vie 22/01/10 | mar 16/02/10 | 5 |
| 8 | **Desarrollo de pruebas** | 6d | mar 16/02/10 | mié 24/02/10 | 7 |
| 9 | **Desarrollo de la documentación** | 41d | mar 02/02/10 | mié 31/03/10 | 6 |

# ORGANIZACIÓN DE LA MEMORIA

La memoria está dispuesta por capítulos para facilitar una lectura más amena. La distribución de éstos es como sigue:

- **CAPÍTULO 1, INTRODUCCIÓN.** En este capítulo se introduce el proyecto y se comenta como se va a distribuir el trabajo así como la motivación, los objetivos, las tareas necesarias para llevarlo a cabo y esta organización.
- **CAPÍTULO 2, ESTADO DEL ARTE.** En esta parte de la memoria se describen las tecnologías necesarias para el desarrollo del proyecto. Se hablará brevemente de HTML, XML y XHTML, y se hará con más detalle de Mono, entorno .NET de Microsoft, C# y Visual C#.
- **CAPÍTULO 3, REQUISITOS.** En este capítulo se enumeran los requisitos del sistema a desarrollar, agrupados por su similitud.
- **CAPÍTULO 4, DISEÑO DEL PROGRAMA.** En este capítulo y como consecuencia de los requisitos anteriores, se analiza las distintas alternativas de diseño, se plantea la alternativa por la que se decanta, y se justifica por qué. Se tratarán los temas del lenguaje de programación y el modo de interacción de html2xhtml con la aplicación.
- **CAPÍTULO 5, IMPLEMENTACIÓN.** En esta parte de la memoria se revela aspectos importantes de ciertas partes relevantes de la aplicación para una mejor comprensión del lector. Se hablará, entre otras cosas, de cómo interacciona la aplicación con html2xhtml, como se colorea el código, etc.
- **CAPÍTULO 6, PRUEBAS.** En este capítulo se describe el proceso seguido para comprobar el buen funcionamiento de la aplicación mediante la utilización de diversos archivos con distinto contenido y tamaño.
- **CAPÍTULO 7, CONCLUSIONES Y PLAN DE FUTURO.** En esta parte del proyecto se desarrollan los problemas y soluciones derivados de la utilización del Entorno Mono, la consecución de objetivos y los futuros trabajos a partir de este proyecto.

# ESTADO DEL ARTE

## HTML y XHTML

El lenguaje HTML (Hyper Text Markup Language) no es más que una aplicación de SGML (Standard Generalized Markup Language), un sistema para definir tipos de documentos estructurados y lenguajes de marcas para representar esos mismos documentos. El término HTML se suele referir a ambas cosas, tanto al tipo de documento como al lenguaje de marcas.

A medida que se afianza en el manejo de la Web, se pasa por tres etapas diferentes: Al principio solamente se conocen unas pocas páginas, luego existen buscadores lo cual lo hace más interesante y por último se sabe que en Web no solamente se puede ver la información sino que también se puede publicar.

El HTML es el lenguaje de marcas de texto utilizado normalmente en la WWW (World Wide Web). Fue creado en 1989 por el físico nuclear Tim Berners-Lee; quien tomó dos herramientas preexistentes: el concepto de Hipertexto que permite conectar dos elementos entre sí mediante hipervínculos y anclas; y el SGML (Lenguaje Estándar de Marcación General), que sirve para colocar etiquetas o marcas en un texto que indique como debe verse. HTML no es propiamente un lenguaje de programación como C++, Visual Basic, etc., sino un sistema de etiquetas. HTML no es procesado por compiladores, por lo que algún error de sintaxis que se presente éste no será detectado por el navegador y se visualizará en la forma cómo éste lo entienda.

El entorno para editar HTML puede ser simplemente un editor o procesador de texto, como el que ofrecen los sistemas operativos Windows (Bloc de notas), UNIX (el editor vi o ed) o el que ofrece MS Office (Word). El conjunto de etiquetas que se creen, se deben guardar con la extensión .HTM o .HTML.

Estos documentos pueden ser mostrados por los visores o navegadores de páginas Web en Internet, como Firefox, Opera y Microsoft Internet Explorer.

El HTML Dinámico (DHTML) es una mejora de Microsoft de la versión 4.0 de HTML que le permite crear efectos especiales como, por ejemplo, texto que vuela desde la página palabra por palabra o efectos de transición al estilo de anuncio publicitario giratorio entre página y página.

Un poco de historia: en 1945, el Director de la Oficina de Desarrollo e Investigación Científica (EE.UU.), el Doctor Vannevar Bush, escribió el artículo "As We May Think" para "The Atlantic Online", en que expresaba su preocupación por la ingente cantidad de información que existía y estaba siendo generada, y el poco tiempo y los ineficientes sistemas que había para encontrarla. Así, y basándose en la tecnología existente en aquel entonces, describió un dispositivo personal, al que llamó "memex", y que imaginaba como un suplemento íntimo a su memoria. Este aparato permitiría a cada individuo almacenar su información en microfilmes, consultarlos rápidamente y, lo que es más importante, crear vínculos entre unos documentos y otros, de modo que durante la lectura de un documento se recordara al lector qué documentos contenían información relacionada. Era una visión de lo que ocurriría sólo 45 años después.

En los años 60, Douglas Engelbart, mientras trabajaba en el Stanford Research Institute, propuso el NLS (oNLine System), un entorno de trabajo por computadora, con un sistema para almacenar publicaciones, con catálogos e índices para facilitar la búsqueda, y con reglas establecidas para citar documentos, de modo que fuera más fácil para los lectores acceder a los documentos referenciados. Era un entorno con teclado, vista, ratón e impresora, con posibilidad de tele conferencia y correo electrónico a través de una red de computadoras para una rápida comunicación entre los profesionales. Tenía las herramientas básicas de composición, estudio, organización y modificación de información. Los ficheros se guardaban jerárquicamente para su mejor organización. Se trabajaba con los documentos en modo multiventana, para ver varios documentos a la vez en ventanas diferentes, y se podían copiar objetos seleccionados de una ventana a otra.

El término "hipertexto" fue acuñado por Ted Nelson en 1965, en su artículo "A File Structure for the Complex, the Changing, and the Indeterminate", que leyó durante la vigésima conferencia anual de la Association of Computer Machinery (ACM). Ted Nelson ideó un modelo para la interconexión de documentos electrónicos. El proyecto

Xanadú aún continúa luchando para conseguir un modelo de hipertexto superior al que trajo la World Wide Web.

La World Wide Web fue inventada en 1989 por un físico del CERN (Organización Europea de Investigación Nuclear) llamado Tim Berners-Lee. Era un sistema de hipertexto para compartir información basado en Internet, concebido originalmente para servir como herramienta de comunicación entre los científicos nucleares del CERN. Tim Berners-Lee había estado experimentando con hipertexto desde 1980, año en que programó *Enquire*, un programa para almacenar piezas de información y enlazarlas entre ellas. *Enquire* se ejecutaba en un entorno multiusuario y permitía acceder a varias personas a los mismos datos. Tim Berners-Lee entregó su propuesta al CERN en 1989, en septiembre de 1990 recibió el visto bueno y junto con Robert Cailliau comenzó a escribir el nuevo sistema de hipertexto. A finales de 1990 el primer *navegador* de la historia, World Wide Web, ya tenía forma.

Los documentos necesitaban un formato que fuera adecuado para su misión. En aquella época casi todo el mundo utilizaba TeX y PostScript, pero éstos eran demasiado complicados teniendo en cuenta que debían ser leídos por todo tipo de computadoras, desde las terminales “tontas” hasta las estaciones de trabajo gráficas X-Windows. Así, tanto el lenguaje de intercambio (HTML), como el protocolo de red (HTTP) se diseñaron para ser realmente muy simples.

El primer documento formal con la descripción de HTML se publicó en 1991 bajo el nombre *"HTML Tags"* (*Etiquetas HTML*) y todavía hoy puede ser consultado online a modo de *reliquia informática*.

La primera propuesta oficial para convertir HTML en un estándar se realizó en 1993 por parte del organismo IETF (*Internet Engineering Task Force*). Aunque se consiguieron avances significativos (en esta época se definieron las etiquetas para imágenes, tablas y formularios) ninguna de las dos propuestas de estándar, llamadas HTML y HTML+ consiguieron convertirse en estándar oficial.

En 1995, el organismo IETF organiza un grupo de trabajo de HTML y consigue publicar, el 22 de septiembre de ese mismo año, el estándar HTML 2.0. A pesar de su nombre, HTML 2.0 es el primer estándar oficial de HTML.

A partir de 1996, los estándares de HTML los publica otro organismo de estandarización llamado W3C (*World Wide Web Consortium*). La versión HTML 3.2 se publicó el 14 de Enero de 1997 y es la primera recomendación de HTML publicada por el W3C. Esta revisión incorpora los últimos avances de las páginas Web desarrolladas hasta 1996, como *applets* de Java y texto que fluye alrededor de las imágenes.

HTML 4.0 se publicó el 24 de Abril de 1998 (siendo una versión corregida de la publicación original del 18 de Diciembre de 1997) y supone un gran salto desde las versiones anteriores. Entre sus novedades más destacadas se encuentran las hojas de estilos CSS, la posibilidad de incluir pequeños programas o *scripts* en las páginas Web, mejora de la accesibilidad de las páginas diseñadas, tablas complejas y mejoras en los formularios.

La última especificación oficial de HTML se publicó el 24 de diciembre de 1999 y se denomina HTML 4.01. Se trata de una revisión y actualización de la versión HTML 4.0, por lo que no incluye novedades significativas.

Desde la publicación de HTML 4.01, la actividad de estandarización de HTML se detuvo y el W3C se centró en el desarrollo del estándar XHTML. Por este motivo, en el año 2004, las empresas Apple, Mozilla y Opera mostraron su preocupación por la falta de interés del W3C en HTML y decidieron organizarse en una nueva asociación llamada WHATWG (*Web Hypertext Application Technology Working Group*).

La actividad actual del WHATWG se centra en el futuro estándar HTML 5, cuyo primer borrador oficial se publicó el 22 de enero de 2008. Debido a la fuerza de las empresas que forman el grupo WHATWG y a la publicación de los borradores de HTML 5.0, en marzo de 2007, el W3C decidió retomar la actividad estandarizadora de HTML.

Hoy, en 2009, la Web es algo cotidiano para una gran parte de los más de 800 millones de usuarios de Internet que hay en todo el mundo. Sus utilidades son diversas, su impacto en la economía mundial es apreciable. No sólo hay documentos de texto: hay imágenes, vídeos, música, se pueden comprar cosas, se pueden hacer reservas...

De forma paralela a su actividad con HTML, W3C ha continuado con la estandarización de XHTML, una versión *avanzada* de HTML y basada en XML. La

primera versión de XHTML se denomina XHTML 1.0 y se publicó el 26 de Enero de 2000 (y posteriormente se revisó el 1 de Agosto de 2002).

XHTML 1.0 es una adaptación de HTML 4.01 al lenguaje XML, por lo que mantiene casi todas sus etiquetas y características, pero añade algunas restricciones y elementos propios de XML. La versión XHTML 1.1 es ya una recomendación (al igual que XHTML Basic 1.1) y pretende modularizar XHTML. También ha sido publicado el borrador de XHTML 2.0, que supondrá un cambio muy importante respecto de las anteriores versiones de XHTML, aunque actualmente parece que su desarrollo esté ya abandonado.

El lenguaje XHTML es muy similar al lenguaje HTML. De hecho, XHTML no es más que una adaptación de HTML al lenguaje XML. Técnicamente, HTML es descendiente directo del lenguaje SGML, mientras que XHTML lo es del XML (que a su vez, también es descendiente de SGML).

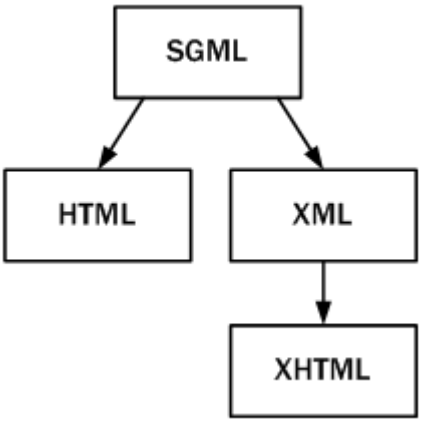



**Figura 2. Esquema de la evolución de HTML y XHTML**

Las páginas y documentos creados con XHTML son muy similares a las páginas y documentos HTML. Las discusiones sobre si HTML es mejor que XHTML o viceversa son recurrentes en el ámbito de la creación de contenidos Web, aunque no existe una conclusión ampliamente aceptada.

Actualmente, entre HTML 4.01 y XHTML 1.0, la mayoría de diseñadores escogen XHTML. En un futuro cercano, si los diseñadores deben elegir entre HTML 5 y

XHTML 1.1 o XHTML 2.0, quizás la elección sea diferente. En cualquier caso, HTML 5 también define una notación basada en XML.

# ENTORNOS GRÁFICOS DE USUARIO (GUI'S)

Los usuarios están tan acostumbrados a trabajar con los entornos gráficos que, para la mayoría, constituyen la única experiencia que tienen con sus ordenadores. Sin embargo, las 'carpetas' y 'ventanas' no son, ni mucho menos, el lenguaje innato de un ordenador.

El entorno de escritorio (en inglés, 'desktop environment') es el conjunto de iconos, barras de herramientas, fondo de pantalla y habilidades específicas que llamamos 'escritorio' en el sistema operativo. La información se manipula directamente sobre las representaciones gráficas con ayuda del puntero y el ratón.

El ordenador es una máquina que procesa y almacena información en código binario. Antes de que existieran los dispositivos de almacenamiento, los programadores tenían que escribir, línea por línea, los comandos necesarios para conseguir ejecutar el proceso deseado.

Gracias a los discos duros y los compiladores, los ordenadores de hoy no se programan sino que se utilizan: cada vez que el usuario abre una aplicación está invocando un proceso que alguien ha programado de antemano. El sistema operativo es el conjunto de programas que permiten gestionar dichas aplicaciones y, el entorno gráfico, la representación metafórica de dicho sistema.

Los ordenadores modernos tienen unas orejas para escuchar al usuario (generalmente, un teclado y un ratón) y una boca para contestarle (un monitor o pantalla). En los entornos e interfaces de texto o de Línea de Comandos (CLI), el usuario 'habla' con el ordenador mediante una serie de expresiones llamadas comandos, que son reconocidas por el sistema. Si la orden es correcta, la respuesta del ordenador es la ejecución del proceso. Si no lo es, la pantalla despliega un mensaje de error.

En las interfaces gráficas, el usuario habla y escucha a través de metáforas, un lenguaje visual que representa comandos, procesos y programas en forma de carpetas, iconos y barras de herramientas. Los más conocidos son los escritorios de Windows (Microsoft) y Mac OSX (Apple). En los entornos de software libre, los más populares son GTK+ (Gnome) y Qt (KDE).

La historia de la interfaz (o entorno) gráfica para un sistema operativo, con iconos y un cursor dirigido por un ratón, comenzó con los diseños de Douglas Englebart en 1968, pero la primera compañía en lanzarla al mercado fue Xerox en 1981 con su Sistema de Información Estrella 8010 o 'Xerox Star'.

Ellos inventaron el lenguaje visual que utilizan la mayor parte de los ordenadores personales: carpetas, ventanas, puntero, etc. Apple recogió dicha idea y popularizó el sistema con 'Apple Lisa' en enero de 1983. Lisa introdujo conceptos brillantes que todavía utilizamos hoy, como el modo de navegar por el sistema de archivos de manera jerárquica y el 'drag&drop' (arrastrar y soltar) para mover documentos o copiarlos de unas carpetas a otras.

El ordenador fue un desastre estrepitoso, debido principalmente a su precio, pero el entorno gráfico no. Le siguió VisiOn, de VisiCorp, para ordenadores IBM a mediados de 1983, Macintosh en 1984, la primera versión de Microsoft Windows en 1985 y muchos otros. En 1987, Apple inició un pleito contra Microsoft y Hewlett-Packard por infracción de propiedad intelectual sobre el concepto de entorno gráfico, una guerra que perdió.

En los años 90, Windows se constituyó como el sistema operativo más popular del planeta. En l998, el mundo del software libre (históricamente asociado a los entornos de texto) aportó dos proyectos de entorno gráfico de usuario: Gnome y KDE. Junto con Mac OSX, introducido en el 2002, éstas son las interfaces gráficas alternativas a Windows más populares del mundo.

Los entornos gráficos han sido determinantes a la hora de extender el uso de ordenadores a la población no especializada. Permiten arrastrar documentos, manipular la información o agregar datos en el sistema de manera rápida e intuitiva (es más fácil

recordar un proceso asociado a una imagen que recordar el nombre de dicho proceso, a veces un poco abstracto y normalmente en inglés).

El margen de error es más pequeño, porque hay posibilidades de confundir un comando o deletrearlo mal. Sus detractores, sin embargo, critican su efectividad por dos motivos fundamentales: crean dependencias del usuario a programas y sistemas operativos concretos y obstaculizan el aprendizaje.

Aunque la tendencia habitual entre los diseñadores de entornos gráficos es imitar los entornos más populares para cada sistema operativo o programas especializados, la interfaz gráfica obliga al usuario a aprender una manera específica de manejar su ordenador y le resulta difícil probar nuevos sistemas y programas, porque tiene que desaprender lo aprendido y empezar otra vez. La línea de comandos es la misma en todos los sistemas, los entornos gráficos varían de un sistema a otro, de un programa a otro.

Además, hay quien considera que la interfaz es un muro artificial entre la computadora y su usuario, que obstaculiza en lugar de facilitar la relación entre ambos. El usuario que depende de un entorno gráfico sabe mucho acerca de cómo funciona su interfaz. Aquellos que prescinden de la metáfora, aprenden cómo funciona su ordenador.

# PLATAFORMA .NET

La nueva tecnología de Microsoft ofrece soluciones a los problemas de programación actuales, como son la administración de código o la programación para Internet. Para aprovechar al máximo las características de .Net es necesario entender la arquitectura básica en la que esta implementada esta tecnología y así beneficiarse de todas las características que ofrece esta plataforma.

El Entorno de .Net es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican el desarrollo de aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables. Los principales componentes de este entorno son:

- Lenguajes de programación

- Biblioteca de clases de .Net
- CLR (Common Language Runtime)

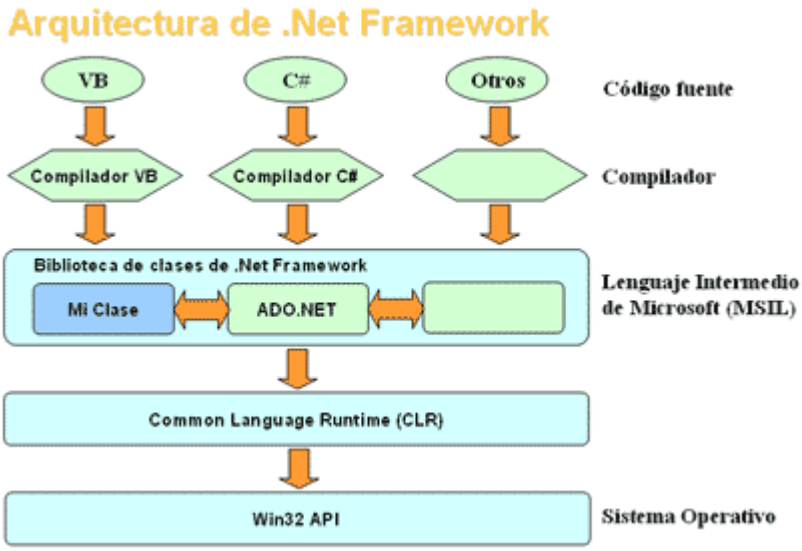



**Figura 3: Arquitectura de .NET Framework**

Actualmente, el Entorno de .Net es una plataforma no incluida en los diferentes sistemas operativos distribuidos por Microsoft, por lo que es necesaria su instalación previa a la ejecución de programas creados mediante .Net. El Entorno se puede descargar gratuitamente desde la Web oficial de Microsoft [13].

El Entorno de .NET soporta múltiples lenguajes de programación y aunque cada lenguaje tiene sus características propias, es posible desarrollar cualquier tipo de aplicación con cualquiera de estos lenguajes. Existen más de 30 lenguajes adaptados a .Net, desde los más conocidos como C# (C Sharp), Visual Basic o C++ hasta otros lenguajes menos conocidos como Perl o Cobol.

## Componentes

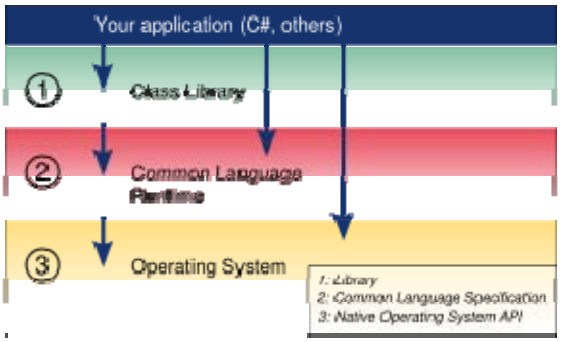



**Figura 4: Componentes de .NET**

## Bibliotecas de clase

Las bibliotecas de clase proveen un conjunto de facilidades que ayudan al desarrollo de aplicaciones. Son escritas primeramente en C#, pero gracias al lenguaje común de especificación (CLS), las mismas pueden ser invocadas en cualquier otro lenguaje de .NET. Cuando hablamos del Entorno de .NET, nos estamos refiriendo primer lugar a las bibliotecas de clase.

## Lenguaje Común de Infraestructura (CLR)

El lenguaje común de infraestructura o más comúnmente llamado **Common Language Runtime** (CLR) es implementado por el ejecutable de Mono. El entorno de ejecución (en inglés, “runtime”) es utilizado para correr aplicaciones compiladas en .NET. Este lenguaje común de infraestructura está definido en los estándares ECMA-335. Para ejecutar una aplicación se deberá invocar el *entorno de ejecución* con los parámetros adecuados.

### Lenguaje Común de Especificación (CLS)

Se encuentra especificado en el estándar ECMA-335 y define la interfase con el CLR. Por ejemplo, convenciones sobre el tipo de datos que se utilizará para implementar los enumerados. El compilador Mono genera una imagen que cumple con el CLS, esta imagen está codificada en el denominado **Common Intermediate Language** (CIL) o **Lenguaje Intermedio Común**. El *runtime* de Mono toma dicha imagen y la ejecuta.

## Biblioteca de clases de .Net

Cuando se está programando una aplicación muchas veces se necesitan realizar acciones como manipulación de archivos, acceso a datos, conocer el estado del sistema, implementar seguridad, etc. El Entorno de .NET organiza toda la funcionalidad del sistema operativo en un espacio de nombres jerárquico de forma que a la hora de programar resulta bastante sencillo encontrar lo que se necesita.

Para ello, el Entorno de .NET posee un sistema de tipos universal, denominado Common Type System (CTS). Este sistema permite que el programador pueda interactuar los tipos que se incluyen en el propio Entorno de .NET (biblioteca de clases de .Net) con los creados por él mismo (clases). De esta forma se aprovechan las ventajas propias de la programación orientada a objetos, como la herencia de clases predefinidas para crear nuevas clases, o el polimorfismo de clases para modificar o ampliar funcionalidades de clases ya existentes.

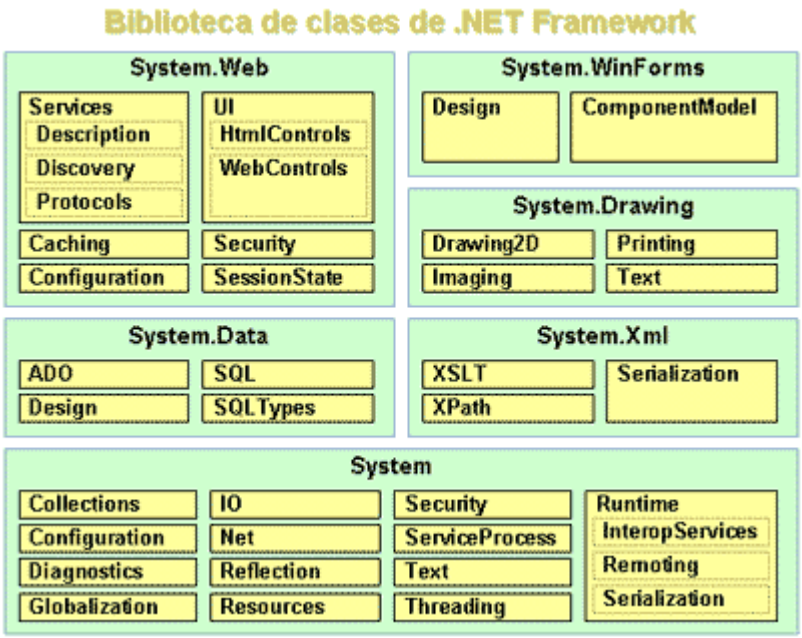



**Figura 5: Bibliotecas de Clase de .NET**

La biblioteca de clases de .Net incluye, entre otros, tres componentes clave:

- ASP.NET para construir aplicaciones y servicios Web.
- Windows Forms para desarrollar interfaces de usuario.
- ADO.NET para conectar las aplicaciones a bases de datos.

La forma de organizar la biblioteca de clases de .Net dentro del código es a través de los espacios de nombres (namespaces), donde cada clase está organizada en espacios de nombres según su funcionalidad. Por ejemplo, para manejar ficheros se utiliza el espacio de nombres System.IO y si lo que se quiere es obtener información de una fuente de datos se utilizará el espacio de nombres System.Data.

La principal ventaja de los espacios de nombres de .Net es que de esta forma se tiene toda la biblioteca de clases de .Net centralizada bajo el mismo espacio de nombres (System). Además, desde cualquier lenguaje se usa la misma sintaxis de invocación, ya que a todos los lenguajes se aplica la misma biblioteca de clases. Los espacios de nombres, por lo general, están compuestos por muchos ensamblados y un ensamblado puede estar compuesto de varios archivos.

## Ensamblados

Uno de los mayores problemas de las aplicaciones actuales es que en muchos casos tienen que tratar con diferentes archivos binarios (DLL´s), elementos de registro, conectividad abierta a bases de datos (ODBC), etc.

Para solucionarlo el Entorno de .NET maneja un nuevo concepto denominado ensamblado. Los ensamblados son ficheros con forma de EXE o DLL que contienen toda la funcionalidad de la aplicación de forma encapsulada. Por tanto la solución al problema puede ser tan fácil como copiar todos los ensamblados en el directorio de la aplicación.

Con los ensamblados ya no es necesario registrar los componentes de la aplicación. Esto se debe a que los ensamblados almacenan dentro de si mismos toda la información necesaria en lo que se denomina el manifiesto del ensamblado. El manifiesto recoge todos los métodos y propiedades en forma de meta-datos junto con otra información descriptiva, como permisos, dependencias, etc.

Para gestionar el uso que hacen la aplicaciones de los ensamblados .Net utiliza la llamada caché global de ensamblados (GAC, Global Assembly Cache). Así, el Entorno de .NET puede albergar en el GAC los ensamblados que puedan ser usados por varias aplicaciones e incluso distintas versiones de un mismo ensamblado, algo que no era posible con el anterior modelo COM.

# WINDOWS FORMS Y VISUAL STUDIO.NET

La creación de aplicaciones Windows (o aplicaciones de escritorio) ha resultado siempre una tarea compleja debido a la dificultad de tener que crear una interfaz gráfica que interactúe con el usuario. Los *Formularios de Windows* (Windows Forms) de .Net permiten la creación de aplicaciones de interfaz gráfica de forma sencilla. .Net proporciona un amplio conjunto de controles como cajas de texto, etiquetas, etc. que, unidos a la completa biblioteca de clases de .Net, hace posible el desarrollo de aplicaciones en poco tiempo.

Para este cometido, Microsoft creo en 1997 Visual Studio en su versión 5.0. Incluía Visual Basic 5.0 y Visual C++ 5.0, para programación en Windows principalmente; Visual J++ 1.1 para programación en Java y Windows; y Visual FoxPro 5.0 para programación en xBase. Introdujo Visual Interdev para la creación dinámica de sitios Web mediante ASP (Active Server Pages). Se incluía una réplica de la biblioteca de código Microsoft Developer Network a modo de documentación.

Visual Studio 5.0 supuso el primer intento de Microsoft para que varios lenguajes utilizaran el mismo entorno de desarrollo. Visual C++, Visual J++, Interdev y MSDN Library hacían uso de un único entorno, denominado Developer Studio. Por otro lado, Visual Basic y Visual FoxPro usaban diferentes entornos.

La siguiente versión, la 6.0, se lanzó en 1998 y fue la última versión en ejecutarse en la plataforma Win9x. Los números de versión de todas las partes constituyentes pasaron a 6.0, incluyendo Visual J++ y Visual InterDev que se encontraban en las versiones 1.1 y 1.0 respectivamente. Esta versión fue la base para el sistema de desarrollo de Microsoft para los siguientes 4 años, en los que Microsoft migró su estrategia de desarrollo al Entorno .NET.

Visual Studio 6.0 fue la última versión en que Visual Basic se incluía de la forma en que se conocía hasta entonces; versiones posteriores incorporarían una versión muy diferente del lenguaje con muchas mejoras, fruto de la plataforma .NET. También supuso la última versión en incluir Visual J++, que proporcionaba extensiones de la plataforma Java, lo que lo hacía incompatible con la versión de Sun Microsystems. Esto acarreó problemas legales a Microsoft, y se llegó a un acuerdo en el que Microsoft

dejaba de comercializar herramientas de programación que utilizaran la máquina virtual de Java.

Aunque el objetivo a largo plazo de Microsoft era unificar todas las herramientas en un único entorno, esta versión en realidad añadía un entorno más a Visual Studio 5.0: Visual J++ y Visual Interdev se separaban del entorno de Visual C++, al tiempo que Visual FoxPro y Visual Basic seguían manteniendo su entorno específico.

Visual Studio .NET puede usarse para crear programas basados en Windows (usando **Windows Forms** en vez de COM), aplicaciones y sitios Web (ASP.NET y servicios Web), y dispositivos móviles (usando el .NET Compact Framework).

Actualmente la versión más utilizada de esta aplicación es Visual Studio 2005, en la que se incluye ".NET framework" en su versión 2.0. La actualización más importante que recibieron los lenguajes de programación fue la inclusión de *tipos genéricos*, similares en muchos aspectos a las plantillas de C#. Con esto se consigue encontrar muchos más errores en la compilación en vez de en tiempo de ejecución, incitando a usar comprobaciones estrictas en áreas donde antes no era posible. C++ tiene una actualización similar con la adición de C++/CLI como sustituto de C# manejado.

**C#**

C# (leído en inglés "C Sharp" y en español "C Almohadilla") es el nuevo lenguaje de propósito general diseñado por Microsoft para su plataforma .NET.

Aunque es posible escribir código para la plataforma .NET en muchos otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado en ella, por lo que programarla usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes ya que C# carece de elementos heredados innecesarios en .NET. Por esta razón, se suele decir que C# es el **lenguaje nativo de .NET**

# CARACTERÍSTICAS DEL LENGUAJE C#

Aunque es pronto para entrar con detenimiento en el lenguaje C# podemos adelantar las características más relevantes de este lenguaje.

- **Es autocontenido**. Un programa en C# no necesita de ficheros adicionales al propio código fuente, como los ficheros de cabecera (.h) de C++, lo que simplifica la arquitectura con respecto a los proyectos software desarrollados con C++.
- **Es homogéneo**. El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina en la que se compile (no ocurre lo que en C++), lo que facilita la portabilidad del código.
- **Es actual**. C# incorpora en el propio lenguaje elementos que se han demostrado ser muy útiles para el desarrollo de aplicaciones como el tipo básico decimal que representa valores decimales con 128 bits, lo que le hace adecuado para cálculos financieros y monetarios, incorpora la instrucción foreach, que permite una cómoda iteración por colecciones de datos, proporciona el tipo básico string, permite definir cómodamente **propiedades** (campos de acceso controlado), etc.
- **Está orientado a objetos**. C# soporta todas las características propias del paradigma de la programación orientada a objetos: *encapsulación*, *herencia* y *polimorfismo*.
  - Encapsulación: además de los modificadores de acceso convencionales: public, private y protected, C# añade el modificador internal, que limita el acceso al proyecto actual.
  - C# sólo admite herencia simple.
  - Todos los métodos son, por defecto, *sellados*, y los métodos redefinibles han de marcarse, obligatoriamente, con el modificador virtual.
- **Delega la gestión de memoria**. Como todo lenguaje de .NET, la gestión de la memoria se realiza automáticamente ya que tiene a su disposición el recolector de basura del CLR. Esto hace que el programador se desentienda de la gestión directa de la memoria (petición y liberación explícita) evitando que se cometan los errores habituales de este tipo de gestión en C++, por ejemplo.

En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad. No permite el uso de punteros, por ejemplo. Sin embargo, y a diferencia de Java, en C# es posible esquivar dichas restricciones manipulando objetos a través de

punteros. Para ello basta marcar regiones de código como inseguras (modificador unsafe) y podrán usarse en ellas punteros de forma similar a como se hace en C++, lo que puede resultar útil para situaciones donde se necesite una eficiencia y velocidad procesamiento muy grandes.

- **Emplea un sistema de tipos unificado**. Todos los tipos de datos (incluidos los definidos por el usuario) siempre derivarán, aunque sea de manera implícita, de una clase base común llamada System.Object, por lo que dispondrán de todos los miembros definidos en esta clase. Esto también es aplicable, lógicamente, a los tipos de datos básicos.
- **Proporciona seguridad con los tipos de datos**. C# no admiten ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código.

C# incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente:

- No pueden usarse variables que no hayan sido inicializadas.
- Sólo se admiten conversiones entre tipos compatibles
- Siempre se comprueba que los índices empleados para acceder a los elementos de una tabla (vector o matriz) se encuentran en el rango de valores válidos.
- Siempre se comprueba que los valores que se pasan en una llamada a métodos que pueden admitir un número indefinido de parámetros (de un cierto tipo) sean del tipo apropiado.

- **Proporciona instrucciones seguras**. En C# se han impuesto una serie de restricciones para usar las instrucciones de control más comunes. Por ejemplo, toda condición está controlada por una expresión condicional, los casos de una instrucción condicional múltiple (switch) han de terminar con una instrucción break o goto, etc.
- **Facilita la extensibilidad de los operadores**. C# permite redefinir el significado de la mayoría de los operadores -incluidos los de conversión, tanto para conversiones implícitas como explícitas- cuando se aplican a diferentes tipos de objetos.
- **Permite incorporar modificadores informativos sobre un tipo o sus miembros**. C# ofrece, a través del concepto de **atributos**, la posibilidad de añadir, a los metadatos del módulo resultante de la compilación de cualquier fuente, información

sobre un tipo o sus miembros a la generada por el compilador que luego podrá ser consultada en tiempo ejecución a través de la biblioteca de **reflexión** de .NET. Esto, que más bien es una característica propia de la plataforma .NET y no de C#, puede usarse como un mecanismo para definir nuevos modificadores.

- **Facilita el mantenimiento (es "versionable")**. C# incluye una política de versionado que permite crear nuevas versiones de tipos sin temor a que la introducción de nuevos miembros provoquen errores difíciles de detectar en tipos hijos previamente desarrollados y ya extendidos con miembros de igual nombre a los recién introducidos.
- **Apuesta por la compatibilidad**. C# mantiene una sintaxis muy similar a C++ o Java que permite, bajo ciertas condiciones, incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes.

En resumen, podemos concluir que:

- Es un lenguaje orientado al desarrollo de componentes (módulos independientes de granularidad mayor que los objetos) ya que los **componentes** son objetos que se caracterizan por sus *propiedades*, *métodos* y *eventos* y estos aspectos de los componentes están presentes de manera natural en C#.
- En C# todo son objetos: desaparece la distinción entre tipos primitivos y objetos de lenguajes como Java o C++ (sin penalizar la eficiencia como en LISP o Smalltalk).
- El software es robusto y duradero: el mecanismo automático de recolección de basura, la gestión de excepciones, la comprobación de tipos, la imposibilidad de usar variables sin inicializar y hacer conversiones de tipo (*castings*) no seguras, gestión de versiones, etc. ayudan a desarrollar software fácilmente mantenible y poco propenso a errores.

# PROYECTO MONO

**Mono** es el nombre de un proyecto de código abierto iniciado por Ximian y actualmente impulsado por Novell (tras la adquisición de Ximian) para crear un grupo de herramientas libres, basadas en GNU/Linux y compatibles con .NET según lo especificado por el ECMA.

Mono posee importantes componentes útiles para desarrollar software:

- Una máquina virtual de lenguaje común de infraestructura (CLI) que contiene un cargador de clases, un compilador en tiempo de ejecución (JIT), y unas rutinas de recolección de memoria.
- Una biblioteca de clases que puede funcionar en cualquier lenguaje que funcione en el CLR (Common Language Runtime).
- Un compilador para el lenguaje C#, MonoBasic (la versión para mono de Visual Basic), Java y Python.
- El CLR y el Sistema de tipos común (CTS) permiten que la aplicación y las bibliotecas sean escritas en una amplia variedad de lenguajes diferentes que compilen para byte code. Esto significa que si, por ejemplo, se define una clase que realice una manipulación algebraica en C#, ésta pueda ser reutilizada en cualquier lenguaje compatible con CLI. Puede crear una clase en C#, una subclase en C++ e instanciar esa clase en un programa en Eiffel.
- Un sistema de objetos único, sistema de hilos, bibliotecas de clases y sistema recolector de memoria pueden ser compartidos por todos estos lenguajes.

Según [1], es un proyecto independiente de la plataforma. Actualmente Mono funciona en GNU/Linux, FreeBSD, UNIX, Mac OS X, Solaris y plataformas Windows.

Existe un proyecto similar, llamado Portable.NET, que es parte del proyecto dotGNU.

## Historia

Para los desarrolladores de .NET, el proyecto MONO representa la única alternativa real a la implementación de Microsoft, ofreciéndonos a los programadores la posibilidad de llevar nuestras soluciones más allá de Windows, sin que por ello la productividad se vea afectada.

En la actualidad, pese a que .NET ya no es un recién llegado, la mayoría de profesionales de la informática creen que se trata de la tecnología creada por Microsoft para competir en el mercado de las herramientas de programación. Por otro lado, se comenta que C# es la respuesta que se ingenió Microsoft para hacer frente a Java, y ni que decir que cuando se habla de .NET hablamos de Microsoft. Quizás sea bueno hacer algunas matizaciones. Ciertamente Microsoft se ha convertido en el principal impulsor e ideólogo de la tecnología .NET pero no ha sido su único creador, ni es la única opción disponible en el mercado. IBM o Sun también han tenido su papel en las actuales especificaciones de los pilares tecnológicos de .NET y del lenguaje C#, y que existen opciones alternativas a la de Microsoft, que permiten por ejemplo ejecutar aplicaciones en Linux o Macintosh.

¿Si Microsoft no ha creado a solas .NET, quién ha colaborado también? En agosto del año 2000 se reunieron Microsoft, Hewlett-Packard e Intel para crear las bases de la tecnología .NET, para posteriormente ir revisándolas y ampliándolas con el esfuerzo en mayor o menor grado de las siguientes empresas e instituciones: Borland, Fujitsu Software, IBM, ISE, IT University of Copenhagen, Jagger Software, Monash University, Netscape, Novell/Ximian, Phone.com, Plum Hall, Sun Microsystems y Univeristy of Canterbury. Todo el trabajo realizado ha sido reunido y estandarizado por la europea ECMA Internacional, concretando dos estándares libres. Por un lado se creó el estándar ECMA-335[10] que sentaba las bases de la máquina virtual y del CLI (*Common Language Infrastructure*) que define las bases de la tecnología .NET y los requisitos que deben cumplir los lenguajes de programación que quieran encajar en ellas, y del otro lado, se redactó el estándar ECMA-334[9] con las especificaciones de C#, a modo de ejemplo o modelo a seguir en cuanto a lenguajes de programación adaptados a la filosofía y arquitectura .NET.

Bajo las pautas de los dos estándares ECMA, Microsoft cogió las especificaciones de .NET y las implementó, y una vez obtenidas las bases de la arquitectura, reprogramó las principales bibliotecas usadas por las herramientas de desarrollo que tenía la compañía de Redmond hasta esa fecha. El fruto de este trabajo dio como resultado bibliotecas tan famosas como ASP.NET o ADO.NET, y creó versiones adaptadas a .NET de algunos lenguajes de programación que ya comercializaba, como por ejemplo, *Visual Basic.NET* o *J#.*

## Los inicios de MONO

En un ambiente como el del software libre, dónde existen infinidad de iniciativas y proyectos, es muy difícil poner a todo el mundo de acuerdo a la hora de escoger un lenguaje de programación que guste a todo el mundo. Es aquí donde las especificaciones de .NET parecen estar hechas a medida para esta situación; Los programadores podrán usar su lenguaje de programación favorito sin tener que hacer ningún esfuerzo a la hora de integrar sus soluciones con otras programadas con diferentes lenguajes. Esta característica, unida a las excelentes especificaciones de .NET le convertían en la opción más productiva.

Una vez que se llega a la conclusión de que la plataforma .NET era la mejor tecnología de desarrollo del momento, se buscó la mejor implementación de la misma, con un resultado obvio, Microsoft .NET. Al igual que en su día se hizo con el UNIX de AT&T, reimplementándolo por completo dando como resultado Linux, se trataba de programar una versión libre de Microsoft .NET. Es importante insistir en este punto: MONO no pretende únicamente realizar una implementación libre del estándar ECMA, como por ejemplo el proyecto dotGNU (*http://www.dotgnu.org*), sino que trata de ir un paso más allá, implementando también las bibliotecas de código que ha producido Microsoft sobre la nueva tecnología, como por ejemplo las ya comentadas ADO.NET y ASP.NET.

## Más allá de los límites de Microsoft

Al igual que en su día había hecho Microsoft, reprogramando sus bibliotecas con la nueva tecnología .NET, los desarrolladores de MONO decidieron hacer lo mismo con

las que estaban usando ellos mismos, como por ejemplo GTK+, que proporciona herramientas para crear aplicaciones de escritorio, creando una nueva versión denominada GTK#. Esta biblioteca ha sido el pilar a la hora de desarrollar las aplicaciones de escritorio mediante MONO dejando hasta hace poco de lado la tecnología usada por Microsoft para estos menesteres: *WinForms*. No fue GTK# ni la primera ni la última, y en la actualidad MONO implementa una rica variedad de bibliotecas no presentes en su homónima de Microsoft. Podremos ver más detalle de las mismas en la página Web de MONO [14].

El proyecto MONO también dio otro gran paso que por cuestiones comerciales, al menos hasta la fecha, Microsoft no ha querido realizar, implementar la máquina virtual de .NET ya no sólo para Windows, sino que también para un amplio espectro de entornos que en la actualidad se puede resumir con Windows, Linux, Solaris, Mac OS X y UNIX.

Llegados a este punto se puede concluir que MONO no sólo ofrece lo que ya brinda Microsoft, además existe una serie de bibliotecas muy útiles para trabajar en sistemas operativos que no sean Microsoft Windows. Pero lamentablemente la situación actual no es tan idílica, aunque tampoco es nada mala. Pese a que la intención es implementar todas bibliotecas de código de Microsoft, de momento se ha quedado en eso, en intención, aunque el porcentaje de bibliotecas cubiertas no es para nada desdeñable, y en breve se tratará de ver cuál ha sido la evolución, el estado en que se encuentra el proyecto, y sus perspectivas de futuro.

### “Write once, run anywhere”

Son muchos los desarrolladores que han entrado en discusiones técnicas comparando .NET con otras alternativas, y especialmente con Java. A los más hábiles no les han faltado argumentos para ir arrinconando a sus *contrincantes*, pero en muchas ocasiones cuando han visto cerca su *triunfo* han tenido que ver frustradas sus expectativas ante la gran ventaja que todo buen desarrollador Java tiene, el famoso: *Escríbelo una vez, ejecútalo dónde quieras*. Dicha afirmación, sobre todo en entornos de servidor, ha decantado muchas veces la balanza hacia el lado de Java, y lamentablemente, en algunas de estas ocasiones ha sido por el desconocimiento de MONO que todavía existe en la comunidad .NET. La filosofía multiplataforma es

inherente a las especificaciones de .NET, y el hecho de que la versión más popular, la de Microsoft, no implemente máquinas virtuales para sistemas operativos no Windows, como ya se ha comentado anteriormente, responde únicamente a cuestiones comerciales, con lo que gracias a MONO ahora un desarrollador de .NET ya puede sentirse tranquilo, la ventaja de Java disminuye.

## ¿Quién hay detrás del proyecto?

En febrero de 2002 cuando Microsoft ya empezaba a mostrar sus cartas entorno a .NET, Miguel De Icaza cofundador del proyecto Gnome[11] dedicado a implementar un gestor de escritorio ágil y potente para Linux, exponía en una lista de correo del proyecto su visión respecto a la tecnología .NET. La intención de Icaza y otros muchos compañeros de fatigas e inquietudes era impulsar a Linux en el mercado de los sistemas operativos de escritorio, donde a diferencia que en los servidores, su presencia era casi testimonial. Gnome empezaba a ser ya un entorno de escritorio amigable y ágil, pero que pese a ofrecer buenos resultados, se estaba convirtiendo en un proyecto cada vez más complicado de mantener y programar, ya que se implementaba en su mayoría con C++ mediante una programación compleja a muy bajo nivel. La necesidad de una herramienta que les proporcionara mayor productividad, los mismos resultados con la mitad de líneas, empezaba a resultar algo realmente imperioso, y .NET empezaba a llamar la atención.

Si contemplamos las fechas de las que estamos hablando veremos que más allá de C++ no existía ninguna herramienta de programación libre que ofreciera la productividad que se estaba buscando. A alguien le puede venir a la cabeza Java, que recientemente ha sido liberado a la comunidad, pero que cabe recordar que en aquellos entonces era un producto con licencias Sun mucho más restrictivas que las actuales. También es cierto que ya existían alternativas libres con implementaciones parciales de Java, y que lamentablemente carecían de coordinación y unificación de esfuerzos.

MONO fue impulsado desde la empresa que éste creo junto a Nat Friedman, Ximian. Pese a esta circunstancia, por la naturaleza del proyecto, en ningún momento MONO fue considerado como un producto de Ximian, sino que desde un principio ha pertenecido a la comunidad de desarrolladores que han ayudado a impulsarlo, o que simplemente han optado por usarlo, todos ellos de forma libre y gratuita.

Posteriormente Novell se interesó por el trabajo que realizaban en la empresa de Icaza y Friedman, y finalmente optó por adquirirla. Evidentemente con la compra Novell no se hizo con la propiedad de MONO ya que como se ha comentado este no pertenecía a Ximian, pero pese a ello, Novell sí ha querido participar activamente en el proyecto, convirtiéndose en el principal patrocinador de la iniciativa, y comercializando servicios relacionados a MONO.

## El camino de MONO

La primera versión del proyecto (Mono v1.0), bautizada con el nombre en clave de "*T-Bone"* (Chuleta), vio la luz en Julio de 2004. La comunidad de desarrolladores disponía ya de una versión completamente funcional y productiva. Además de cubrir las especificaciones ECMA, se implementaron la biblioteca de acceso a datos ADO.NET y la de desarrollo de aplicaciones Web, ASP.NET. En cuanto al desarrollo de aplicaciones de escritorio, se apostó por GTK#, dejando de lado la apuesta tecnológica de Microsoft en este campo, los *WinForms*. MONO en esta primera versión únicamente implementó un compilador para C#, el especificado por ECMA, dejando a terceros la posibilidad de producir otros compiladores.

El pasado noviembre de 2006, apareció la versión 1.2, en esta ocasión bautizada como *Rump Steak* (*Filete*). En esta ocasión el proyecto le tiende una mano a la amplia comunidad de desarrolladores que usan Visual Basic de forma habitual, implementando un compilador de este lenguaje, aunque con algunas características incompletas. Características que sí se implementan por completo en el compilador de C#, el cual incluye la mayor parte de las novedades incluidas en .NET 2.0. Por otro lado se echa otro *capote* al gran número de programadores que tienen implementadas soluciones usando *WinForms*. Al igual que con Visual Basic, la implementación de esta tecnología para esta versión es parcial, quedando todavía mucho trabajo de cara al futuro si se pretende conseguir una compatibilidad cercana al 100%.

A finales de 2008 aparece Mono v2.0. Siguiendo la línea de anteriores versiones ha sido bautizada como *Sirloin* (Solomillo). La nueva versión Mono 2.0 es compatible con los componentes de servidor y escritorio de la versión 2.0 de .NET.

Mono 2.0 ofrece a los desarrolladores de .NET la libertad de hacer funcionar sus aplicaciones en una amplia variedad de sistemas operativos, incluidos Linux, Mac OS y Unix. La versión Mono 2.0 beneficia a un gran número de desarrolladores, usuarios finales a los que permite escribir sus aplicaciones una sola vez y utilizarlas en cualquier sistema operativo, incrementando notablemente la portabilidad y ampliando su alcance en el mercado.

El nuevo Mono 2.0 incorpora la herramienta Mono Migration Analyzer (MoMA), que funciona tanto sobre la plataforma .NET como sobre Mono, permite a los desarrolladores cuantificar el número de cambios requeridos para ejecutar sus aplicaciones .NET en un entorno Linux. Un análisis realizado a 4.600 aplicaciones .NET que usan MoMA revela que el 45% de ellas no han requerido cambios en el código para trabajar con Mono, mientras que un 24% ha necesitado menos de seis cambios de código para funcionar en Mono.

El proyecto Mono permite el desarrollo multiplataforma. Uno de los usos más exitosos de Mono es el rápido desarrollo de MoonlightTM, una versión plug-in de Microsoft Silverlight, de código abierto y basada en Mono, que se usa para crear y albergar aplicaciones muy interactivas de nueva generación.

Mono 2.0 racionaliza el desarrollo de aplicaciones basadas en .NET Las nuevas funciones de Mono 2.0 incluyen:

- Fácil instalación. Función de instalación con un solo clic para SUSE Linux Enterprise y openSUSE, así como instaladores fáciles de usar para muchas de las otras plataformas soportadas, incluidas Windows y Mas OS X.
- Soporte de plataforma integral, API y hardware. El Entorno Mono soporta numerosas plataformas, incluidas Linux, Mac OS X, Solaris, BSD y Windows; una variedad de opciones de hardware, tales como x86, AMD 64, IA-64 (Itanium 2), EMT 64, PowerPC, ARM, S390 y S390x, SPARC y SPARC 9; todas las APIs de Microsoft .NET 2.0, incluidas ASP.NET, ADO.NET y Windows.Forms; y el compilador C# 3.0 con soporte Language Integrated Query (LINQ).
- Incremento del rendimiento. Mejora la escala y rendimiento para ASP.NET, ADO.NET y el tiempo de ejecución Mono.

- Descargas útiles. Una imagen de máquina virtual con un entorno de desarrollo listo para usar y numerosas aplicaciones .NET de sobremesa y Web de código abierto, incluidas ASP.NET Starter Kits y otras demos. También está disponible la versión actualizada de la herramienta MoMA, con cobertura mejorada

# Las herramientas de desarrollo

El desarrollador acostumbrado a programar para Windows podrá encarar proyectos para, por ejemplo, Linux o Mac del mismo modo que venía haciéndolo hasta la fecha, desarrollando con el avanzado entorno *Visual Studio*. Existen alternativas a pagar el coste de la licencia de Visual Studio, como su variante "Express", que es gratuita, o algunas alternativas de código libre. Dichas alternativas se encuadran en el uso de un sistema operativo gratuito, que en este momento se traduce casi automáticamente en Linux, y en el entorno de desarrollo gratuito de MONO, el *MonoDevelop*. Pese a ser un entorno de desarrollo bastante potente y profesional, está muy lejos todavía de alcanzar los niveles de *Visual Studio* incluso en sus versiones Express. Incluye funcionalidades básicas como coloreado de código, autocompletado de código y desde hace poco tiempo seguimiento de procesos mediante un depurador. En cuanto al diseño visual de interfaces de usuario el soporte es realmente pobre, cabiendo destacar únicamente los aspectos referidos al diseño mediante GTK#, aunque por ejemplo en *WinForms* es totalmente inexistente.

Como complemento a *MonoDevelop* también existe la aplicación *MonoDoc*, la alternativa al MSDN de Microsoft, o en definitiva, la ayuda y documentación necesaria a la hora de abordar un desarrollo con .NET. Al igual que sucede con MSDN existe una versión on-line en: *http://www.go-mono.com/docs/*.

Paralelamente a *MonoDevelop*, y al proyecto MONO en sí, existe otra alternativa libre a *Visual Studio*, *SharpDevelop*[12] que recientemente también da soporte a MONO mucho más rico que *MonoDevelop*, sobre todo en su modo diseñador, pero a diferencia de este, sólo existe una versión para Windows.

La mejor manera de evaluar la madurez de una tecnología es ver sus resultados, y en el caso de una herramienta de desarrollo, qué mejor que ver las soluciones desarrolladas con la misma.

Dado que MONO nació en el ambiente de Gnome, es quizás éste uno de los mejores ejemplos de su uso. Pese a que Gnome no ha sido reescrito con MONO, sí que ha sido su herramienta de crecimiento hasta nuestros días, convirtiéndolo en el sistema de escritorio más usado en Linux.

Siguiendo con el objetivo inicial de potenciar Linux en el escritorio han sido creadas con MONO aplicaciones como: *iFolder,* para la gestión de documentos de forma distribuida, Beagle para potenciar las búsquedas instantáneas dentro de los dispositivos de almacenamiento, *F-Spot*, para gestionar bibliotecas de fotografías e imágenes, *Banshee*, un potente reproductor multimedia, entre otras muchas. El máximo exponente de versatilidad como herramienta de vitalización del escritorio lo encontramos en la gestión tridimensional implementada en XGL, también para Linux, cuyos espectaculares vídeos han invadido Internet

Pero MONO no solo se ha quedado bajo el amparo del escritorio de Linux, sino que está explotando sus características en un campo tan competitivo como el de los videojuegos. En un sistema operativo como Mac OS X, en el que el mercado del juego digital está lejos de ser explotado de forma masiva, se ha presentado una herramienta de sorprendente potencial, *Unity* (*http://unity3d.com/*) con la que podremos diseñar y codificar potentes videojuegos tridimensionales, programando inteligencia artificial mediante C#. Aunque si de videojuegos hablamos, uno de los mayores éxitos de la industria es sin duda *Second* Life (*http://secondlife.com/*), proyecto que ha sido elevado a fenómeno social, y sí, efectivamente, cuenta con MONO para interconectar de forma online las vidas digitales de más de tres millones de usuarios. En la actualidad MONO implementa la totalidad de la lógica del juego, anteriormente escrita en *LSL*, consiguiendo multiplicar el rendimiento de los scripts, usando sólo la mitad de la memoria que venía siendo necesaria hasta la llegada de .NET. En estos momentos la granja de servidores que soportan dicha lógica supera ya los 6.000 servidores, algo que deja en evidencia la madurez, potencia y fiabilidad de MONO.

# Desarrollo en Mono y propiedad intelectual

En el momento que el equipo de desarrollo abandonó el amparo de los estándares ECMA liberados de patentes, ya no sólo para implementar sus propias bibliotecas de códigos, sino también para reimplementar bibliotecas de Microsoft, como ASP.NET, ADO.NET o *WinForms*, el proyecto entró en un terreno legal en el que muchos desarrolladores no se sienten muy cómodos. La implementación de estas bibliotecas creadas por Microsoft no es ilegal, pero se puede incurrir en violaciones de patentes de forma relativamente sencilla, aunque es justo decir que hasta la fecha no ha habido ningún conflicto por este motivo con la empresa de Redmond. La postura oficial de los impulsores del proyecto es muy clara, y se puede resumir en unas pocas palabras: evidentemente tratar por todos los medios no incumplir ninguna patente y en caso de detectarse una violación de patente, buscar usos del algoritmo patentado anteriores a la patente para tratar de invalidarla, o directamente rescribir el código con otra estructura algorítmica que no infrinja dicha patente. Los cambios en caso de que fueran necesarios deberán tratar como última opción romper la interfaz expuestas por las bibliotecas.

Pese a esta relación con Microsoft, la verdad es que cualquier proyecto de software libre de la envergadura de MONO, con miles y miles de líneas de código, difícilmente no va incumplir ninguna patente, en un mundo donde se patenta el más pequeño detalle. Es por ello que los *miedos* en esta materia alrededor de MONO se pueden considerar del mismo modo que en el resto de grandes proyectos de software libre, el cual ya dispone de mecanismos de autoprotección en esta materia.

Se ha escrito mucho sobre las patentes de software, y también sobre el acuerdo al que llegaron recientemente Microsoft y Novell, para no enfrentarse por este tema. Si bien es cierto que son muchos los que han visto en este acuerdo una declaración de intenciones por parte de Microsoft respecto al proyecto MONO, y posiblemente ha conseguido tranquilizar a los más alarmistas, pero debemos recordar que Novell no es el propietario de MONO, con lo que cualquier acuerdo con Microsoft, en lo que respecta a MONO, es algo meramente representativo y poco tangible. Por otro lado también están los que han visto este acuerdo como una *mancha* en el currículum de MONO, ya que la sola insinuación de una vinculación de Microsoft con el proyecto es visto por los más extremistas dentro del *mundillo* del software libre como una auténtica aberración.

En definitiva el proyecto MONO nos está ofreciendo a todos nuevos caminos a explorar. A los usuarios de Windows y de software propietario les brinda la posibilidad de ampliar su mercado y posibilidades, mientras que por otro lado, la comunidad de desarrolladores con las herramientas y conocimientos adecuados a la hora de poder colaborar con proyectos de software libre aumenta de forma exponencial.

El creciente desarrollo con éxito de aplicaciones para Mono sugiere que puede ser viable su uso como alternativa libre o complemento al entorno .NET de Microsoft.

## Aplicaciones con GUI en .NET: ¿Gtk# o Windows Forms?

Desde hace tiempo se discute la *mejor manera de usar Mono* tanto en el escritorio Linux como en Windows. Me refiero al momento de escoger entre Windows Forms y Gtk#, para desarrollar aplicaciones con interfaz gráfica. Los profesionales del software libre dicen que **la mejor opción es usar Mono + Gtk#**. Claro, eso también depende de los gustos como desarrollador y, más importante aún, de las necesidades del proyecto a realizar.

Lo que se mostrará a continuación es *cómo se pueden tomar diferentes caminos*, y aclarar algunas dudas que se plantean. Se tomará como ejemplo a dos programas sencillos: el primero usando Gtk# y corriendo tanto en Linux como Windows, y el otro usando Windows Forms corriendo tanto en Windows como Linux.

¿Qué se necesita para desarrollar y ejecutar aplicaciones que usen Gtk# y Windows Forms? Es necesario (obviamente) tener dicho Entorno instalado; afortunadamente, la mayoría de las distribuciones Gnu/Linux actuales tienen los paquetes necesarios en los repositorios, e incluso vienen con lo necesario para ejecutar aplicaciones desarrolladas con .NET. *.Para compilar aplicaciones C# para mono es necesario utilizar el CSharp Compiler.*

También es posible descargar mono para los más importantes sistemas operativos, incluyendo por supuesto sistemas Microsoft Windows. Para tal caso es posible descargar una versión completa, o tan solo el intérprete + Gtk# para un *Entorno .NET* [4] ya instalado sobre el ordenador.

```
1. //Ejemplo GTK (EjemploGtk.cs)
2. using Gtk;
3. using System;
4. class EjemploGtk {
5.    static void Main()
6.    {
7.       Application.Init ();
8.       //Crear el boton
9.       Button btn = new Button ("Hola monito!");
10.                  //Asignar un evento al evento Clic del boton
11.                  btn.Clicked += new EventHandler (hola);
12.                  //Crear el objeto ventana
13.                  Window ventana = new Window ("Lindo primate!");
14.                  //Asignar un metodo al evento Cerrar ventan
15.                  ventana.DeleteEvent += new DeleteEventHandler (cerrar_ventana);
16.                  //Asigna el ancho y alto de la ventana
17.                  ventana.DefaultWidth = 200;
18.                  ventana.DefaultHeight = 60;
19.                  //Anyadir el boton a la ventana
20.                  ventana.Add (btn);
21.                  //Mostrar la ventana
22.                  ventana.ShowAll ();
23.                  //Ejecutar la aplicación
24.                  Application.Run ();
25.               }
26.               static void cerrar_ventana (object obj, DeleteEventArgs args)
27.               {
28.                  Application.Quit ();
29.               }
30.               static void hola (object obj, EventArgs args)
31.               {
32.                  Console.WriteLine("Lindo monito!");
33.                  Application.Quit ();
34.               }
35.            }
```

Para compilarlo y ejecutarlo se puede usar los siguientes comandos:

```
1.  Mcs -pkg:gtk-sharp-2.0 EjemploGtk.cs
2. mono EjemploGtk.exe
```

El resultado es el mismo tanto en Linux como en Windows:



Este programa hace exactamente lo mismo que el anterior, pero esta vez usando Windows Forms para la interfaz gráfica:

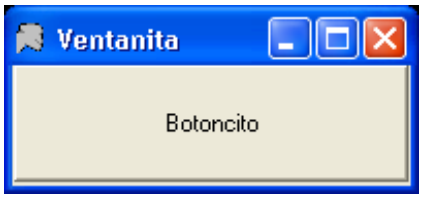
```
1. //Ejemplo Windows Forms (EjemploWinForms.cs)
2. using System.Drawing;
3. using System.Windows.Forms;
4. using System;
5. class EjemploWinForms : Form{
6.    static void Main(){
7.       Application.Run(new EjemploWinForms());
8.    }
9.    public EjemploWinForms() {
10.                  //Crear e iniciar el boton, y sus propiedades
11.                  Button boton = new Button();
12.                  boton.Location = new Point(0, -1);
13.                  boton.Name = "boton";
14.                  boton.Size = new System.Drawing.Size(193, 60);
15.                  boton.TabIndex = 0;
16.                  boton.Text = "Botoncito";
17.                  //Asignar un evento al evento Clic del boton
18.                  boton.Click += new System.EventHandler(hola);
19.                  //Indicar el tamanyo de la ventana
20.                  ClientSize = new System.Drawing.Size(193, 60);
21.                  //Anyadir el boton
22.                  Controls.Add(boton);
23.                  Text = "Ventanita";
24.                  ResumeLayout(false);
25.               }
```

```
26.             private void hola(object sender, EventArgs e) {
27.                 Console.WriteLine("Un mensajito!");
28.                 Close ();
29.             }
30.         }
```

Para compilarlo y ejecutarlo puedes usar los siguientes comandos:

```
1. mcs -r:System.Drawing.dll,System.Windows.Forms.dll EjemploWinForms.cs
2. mono EjemploWindowsForms.exe
```

El resultado es el mismo tanto en Linux como en Windows:


**Consideraciones a tener en cuenta**

Al momento de desarrollar aplicaciones con GUIs en C# de manera rápida, hay básicamente dos opciones:

- **Usar Monodevelop que incluye un editor de interfaces gráficas** basado en Stetic y que por supuesto desarrolla dichas interfaces bajo Gtk#.
- O usar el editor de interfaces de Microsoft Visual Studio que, aunque no es libre, es bastante bueno y completo. En este caso las aplicaciones correrían usando Windows Forms.

El Entorno de Mono incluye bibliotecas para ejecutar aplicaciones que usen Windows Forms. En este sentido la versión 2.0 de Mono permite casi al 100% compilar y ejecutar aplicaciones con bibliotecas Windows Forms en otros sistemas distintos del de Microsoft, con lo cual, la ejecución de las aplicaciones .NET en otros sistemas distintos de los de Windows se realizaría in ningún problema, a diferencia de la versión 1.0 que sí sería necesario una trasformación más profunda del código.

# LA APLICACIÓN HTML2XHTML

**HTML2XHTML** es una aplicación realizada por el profesor Jesús Arias Fisteus y su equipo dentro del **Departamento de Ingeniería Telemática** de la **Universidad Carlos III de Madrid.** Este programa traduce automáticamente documentos en formato HTML, a su correspondiente XHTML (lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas Web). La aplicación ha sido desarrollada en C, y no requiere de bibliotecas externas para su funcionamiento.

El programa permite, desde la línea de comandos, parametrizar algunas opciones como son el tipo de salida de documento XHTML, el número de caracteres por línea, si desea conservar o no los comentarios, etc. Además, si el archivo HTML no está completo o faltan algunas etiquetas, la aplicación se encarga de completar la salida para que sea correcto.

La herramienta **html2xhtml** se ofrece como servicio Web y su código fuente está disponible en lenguaje c que funciona tanto en plataformas GNU/Linux como Windows, aunque en su versión compilada sólo se ofrece para Windows, siendo necesario en otras plataformas compilarlo a partir de su código fuente.

Desde su puesta a disposición de los usuarios, con su liberación de código en Mayo de 2007, se han intentado introducir mejoras sustanciales para un mejor funcionamiento. Las mejoras incluyen desde poder traducir el código a varios tipos de documentos XHTML (XHTML Basic 1.1 y Print en Septiembre de 2007) hasta la introducción de mejoras en el formato (posibilidad de ajustar los caracteres de línea y tabulación en Enero de 2008).

La forma de ejecución de la aplicación es sencilla: en la línea de comandos se ejecuta el comando html2xhtml.exe seguido del archivo a traducir y después una serie de opciones del tratamiento del fichero y su salida, como se observa en el ejemplo siguiente:

```
Html2xhtml.exe archivo.html –t auto –l 80 –b 0
```

Donde:

-t auto: indica al programa el formato del archivo de salida del documento,

-l 80: le indica al programa el número de caracteres por línea, y

-b 0: le indica la longitud de una unidad de tabulación.

Las primeras versiones de html2xhtml funcionan sólo en modo de línea de comandos. Actualmente existe una API de Web con la capacidad de responder a los documentos enviados desde formularios HTML. El documento HTML que desea convertir y parámetros de configuración se reciben en este modo a través de HTTP con el método POST y el multipart/form-data de codificación, según lo definido por RFC 2388[15].

# REQUISITOS DEL SISTEMA

## Requisitos de usuario

- Conocimientos mínimos de informática.
- Conocimientos mínimos de Sistema Windows o Linux necesarios para ejecutar la aplicación.

## Requisitos de Usabilidad

La aplicación debe cumplir las siguientes características:

- **Que sea efectivo**: Se refiere a cómo de bien un sistema hace lo que se supone que debe hacer.
- **Que sea eficiente**: hace referencia a la forma en que un sistema ayuda a los usuarios a llevar a cabo sus tareas.
- **Que sea seguro**: se refiere a que el usuario está protegido de condiciones peligrosas y de situaciones indeseables.
- **Que sea útil**: Se refiere a que el sistema proporciona el tipo de funcionalidades correcta, de manera que el usuario puede hacer lo que necesita y lo que quiere hacer.
- **Que se pueda aprender fácilmente**: hace referencia al esfuerzo que requiere aprender a usar la aplicación.
- **Que sea fácil de recordar cómo se usa**: se refiere al esfuerzo que requiere recordar una aplicación después de que se haya aprendido como se usa y no se haya utilizado durante un tiempo.

## Requisitos de Hardware

La aplicación se debe ejecutar sobre un PC con una configuración mínima de:

- Procesador: Pentium III.
- Memoria: 256Mb.
- Tarjeta de video y monitor color VGA.
- Unidad de disco óptico: CDROM.

# Requisitos del Software

El sistema operativo sobre el que se debe ejecutar la aplicación puede ser:

- Para la aplicación Windows: Windows Xp o superior y tener instalado el .NET Framework 2.0 y/o Mono 2.0.
- Para la aplicación Linux: Cualquier distribución (probado en OpenSuSe y Ubuntu). Además, para esta última opción debe estar instalado en el sistema la aplicación **HTML2XHTML** y Mono 2.0.

# Requisitos funcionales

La aplicación debe ajustarse a los siguientes requisitos funcionales:

- Debe permitir configurar los distintos parámetros del conversor **HTML2XHTML.**
- Debe permitir cargar y convertir múltiples ficheros a la vez.
- Debe poder editar un fichero ya cargado, tanto antes como después de la conversión.

# DISEÑO DEL PROGRAMA

## LENGUAJE DE PROGRAMACIÓN

Para comenzar, se debe tener claro en qué lenguaje de programación hay que programar la aplicación. Afortunadamente, hoy en día, existen multitud de lenguajes y plataformas para llevar a cabo el trabajo. Fundamentalmente nos vamos a centrar en dos opciones: C# y Java. De la primera, ya conocemos sus aspectos más destacados expuestos en el estado del arte. La popularidad de Java es ahora mayor que nunca en los servidores de páginas Web, gracias a las Java Server Pages (JSP). Esta tecnología permite el desarrollo de complejas y potentes aplicaciones que funcionan a través de Internet y se utilizan a través del navegador.

Inicialmente Microsoft dio soporte a Java y desarrolló su propia máquina virtual, sin embargo debido a diversas extensiones que realizó en la plataforma, al margen de las especificaciones de Sun, fue denunciado por éste y perdió en los tribunales.

En las aplicaciones stand-alone (aplicaciones que dibujan sus propias ventanas y no se visualizan a través de un navegador de Internet) Java es menos utilizado. El uso de la JVM (Máquina Virtual de Java, en inglés, Java Virtual Machine), hace que el arranque de los programas sea más lento y que éstos ocupen más memoria. Sin embargo se están realizando continuos avances, tanto en la velocidad de los ordenadores como en la propia JVM, que mitigan estos efectos. En este campo también es previsible que se incremente el uso de Java para todo tipo de aplicaciones.

Una de las direcciones en que Java se está moviendo, es hacia la de compilar los programas a código nativo, es decir, las aplicaciones se ejecutarían directamente por la CPU del ordenador sin la intermediación de la JVM.

Actualmente es posible compilar programas Java relativamente complejos bajo el sistema operativo Linux.

En cuanto a las herramientas de desarrollo de aplicaciones en Java, existen varias, de gran calidad y además gratuitas, entre las cuales desatacan NetBeans

(http://www.netbeans.org) y Eclipse (http://www.eclipse.org). Entre las herramientas de pago destaca JBuilder de Borland.

Como se ha destacado, las aplicaciones bajo Java pueden ser compiladas de manera más o menos sencilla por Linux, por lo que, en principio, esta sería la tecnología a utilizar. Sin embargo se escogió Visual C# para poder investigar y desarrollar la plataforma .NET en Linux mediante el Proyecto Mono (ya que con Java hay suficiente documentación como para hacer aplicaciones que funcionen perfectamente bajo Windows y Linux) y comprobar si es cierto que las aplicaciones creadas en C# pueden ser ejecutadas bajo entornos Linux de manera eficiente.

También se consideró realizar la aplicación en el lenguaje proporcionado por Mono 2.0: Gtk#. Este lenguaje es una "traducción" de C# proporcionado por Microsoft para Linux. Se escogió C# para comprobar si las aplicaciones escritas en un entorno Windows, pueden ejecutarse correctamente en un entorno Linux con ayuda del Proyecto Mono.

# MODELO DE INTERACCIÓN CON LA APLICACIÓN HTML2XHTML

Para poder realizar bien el programa es necesario que éste realice varias funciones que están disponibles en la aplicación **HTML2XHTML.** En este caso hay varias opciones que se pueden considerar: se puede enlazar dinámica o estáticamente una biblioteca con las funciones necesarias para realizar la conversión, o bien, utilizar el archivo del comando ejecutable para realizar esa transformación.

Para evaluar cada una de estas alternativas se explicará a continuación cada una de ellas.

- Biblioteca enlazada al programa. Aquí hay que distinguir dos opciones: Biblioteca Multi-hilo y Mono-hilo. El modo Multi-hilo permite realizar varias conversiones concurrentes en distintos hilos de un mismo proceso. En cambio, en la opción Mono-hilo no lo permite. Esta elección es la más eficiente por varios motivos: el código del conversor se ejecuta en el mismo proceso con simples llamadas a métodos

(no es necesario crear un proceso en cada llamada) y, por otra parte, permite al conversor recibir el documento HTML y devolver el documento XHTML directamente en *buffers* de memoria, sin necesidad de pasar por disco, con lo que disminuye el tiempo de ejecución del conversor.

- Uso del comando ejecutable: Esta es la solución más sencilla. Como ya se tiene el comando ejecutable para Windows y para Linux, lo único que se debería de hacer es llamar al comando para que realice las operaciones oportunas. El comando ejecutable se encargaría de hacer la transformación y pasaría a la aplicación el resultado de ella. La aplicación funciona en concurrencia sin necesidad de ser adaptado, porque cada invocación al conversor se ejecuta en un proceso independiente. Si bien es la alternativa más sencilla.

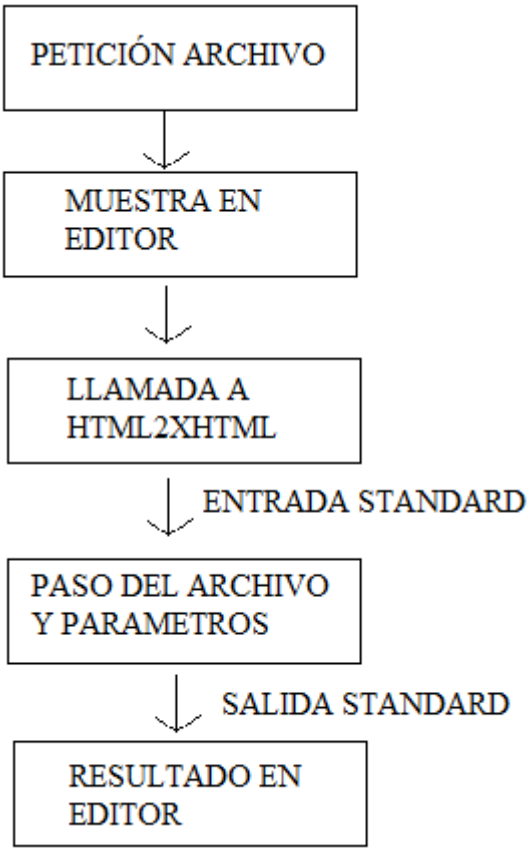
Cualquiera de las dos soluciones anteriores sería válida para este caso. Se elegirá el uso del comando ejecutable por constituir una pérdida de eficiencia mínima y el ahorro del coste del desarrollo es muy grande. Se dejará para futuros proyectos la opción de investigar y construir una biblioteca DLL para enlazarla con la aplicación, ya que la elección de esta solución implicaría mayor costes tanto de tiempo como de personal, ya que implica desarrollar en lenguaje C# la interfaz de comunicación con el conversor, y hacer pruebas exhaustivas para comprobar que el programa html2xhtml limpie adecuadamente la memoria de una invocación a la siguiente, teniendo que corregirlo en caso de que no sea así. Además, en el caso de la opción Multi-hilo la aplicación no está preparada para su funcionamiento con este sistema y se ha estimado que el coste de adaptación sería demasiado elevado.

Para iterar con la aplicación **html2xhtml** se utiliza la entrada/salida estándar de forma que la lectura del archivo y el posterior resultado del conversor se realiza mediante esta entrada sin utilizar archivos temporales. Los parámetros que la aplicación envía al conversor son las siguientes:

- Al conversor se le envía el nombre y directorio donde está alojado el archivo que se va a traducir.
- Se le envía el formato del archivo resultante de la conversión. En el conversor se corresponde con la opción *Output document type* .

- Se envía, así mismo la longitud, en caracteres, de cada línea del documento. Se corresponde con *Line length*
- Además, se envía diversos parámetros opcionales como son el número de tabuladores (*Tab)*, Preservar comentarios (*Preserve spacing in comments)* etc.

En resumen. El funcionamiento de la aplicación es la siguiente:



# ESTRUCTURA DE LA APLICACIÓN

La Aplicación se estructura en tres grandes bloques: Un bloque de inicio, en el que se muestra la vista principal y las distintas opciones disponibles, es la llamada *MDIParent*, la clase destinada la aplicación en sí (donde se convierte el archivo a XHTML), llamado *MDIChild*, y la llamada a la aplicación html2xhtml.

- **Clase MDIParent**. Es la clase principal de la aplicación. Es la encargada de mostrar las distintas opciones de inicio del programa (abrir documento para poder convertirlo, abrir la opción de conversión múltiple…). Esta clase es la encargada de llamar a MDIChild para poder actuar sobre el documento o documentos.
- **Clase MDIChild.** Es la clase donde el usuario puede tratar el archivo para su conversión a cualquiera de los documentos XHTML propuestos. En esta clase se pueden elegir las opciones para la conversión, el tipo de documento de salida entre otras. Se puede, si se desea, modificar manualmente el archivo en el editor, y se puede abrir la opción de múltiples archivos. Como se observa, esta va a ser la clase principal para el usuario puesto que va a poder manipular el documento a su conveniencia.
- **Html2xHhtml.** En realidad no es una clase propiamente dicha. Es la aplicación de intérprete de comandos que nos trasforma el fichero a su equivalente XHTML elegido. La clase anterior es la encargada de formar el comando con los parámetros necesarios y realizar la llamada a esta aplicación, recogiendo los resultados por la salida estándar y mostrándolo al usuario en el editor.

# DISEÑO DE LA INTERFAZ DE USUARIO

En este apartado del proyecto, nos detendremos en las distintas vistas de la aplicación y en cómo el usuario puede interactuar con ellas.

Al inicial la aplicación se observa la vista principal. En ella, se le da la bienvenida al usuario y se presentan las distintas opciones disponibles.

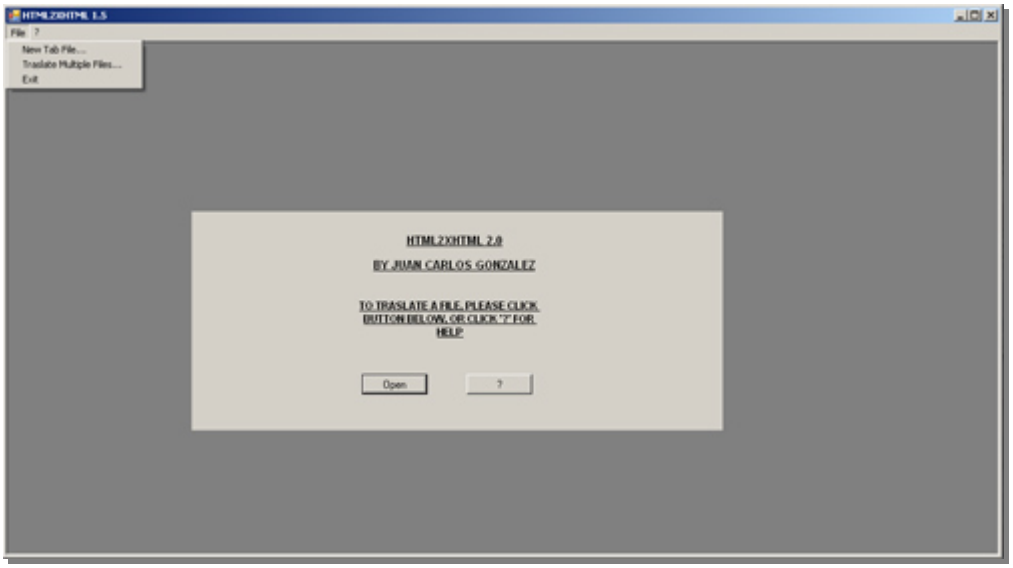



**Figura 6: Vista Principal de la Aplicación.**

Como se observa en la imagen anterior, en la vista principal se pueden realizar diversas actuaciones. Haciendo clic en *Open* se abrirá un cuadro de diálogo que dará la opción de escoger un documento para mostrarlo en el editor.

El botón **?** abrirá la página web de la aplicación html2xhtml de la Universidad Carlos III de Madrid.

Además en el menú *File,* se puede iniciar la aplicación para poder utilizar el convertidor en modo de archivos múltiples con la opción *Translate Multiple Files.* En este mismo menú, aparte de abrir un archivo para la conversión (con la opción *New Tab File)* se puede salir del programa eligiendo *Exit.*

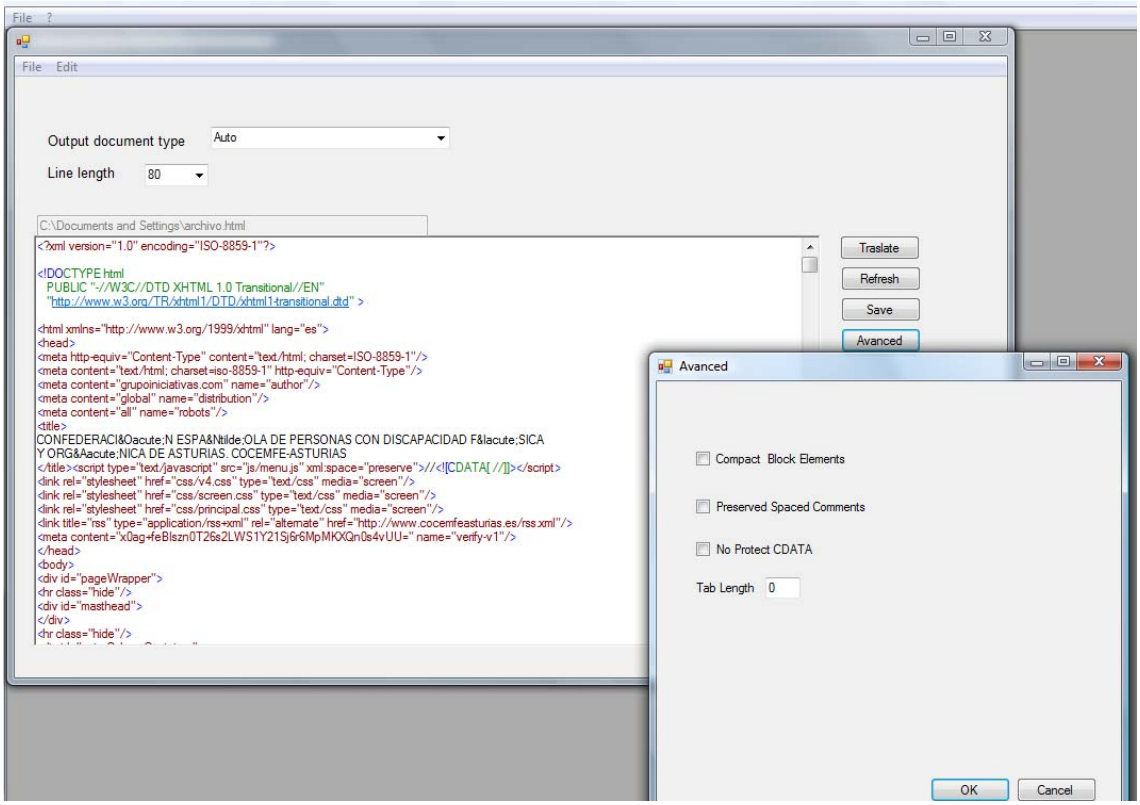



**Figura 7: Vista de edición de archivo con vista de opciones de conversión**

Después de elegir un archivo para la conversión, aparece la vista de la imagen anterior. En ella, se permite la edición del archivo a convertir antes o después de dicha conversión. Además, se permite al usuario cambiar el tipo de salida del archivo así como el tamaño en caracteres de cada línea del documento.

Como se observa, en esta vista hay disponibles diversos botones que hacen más fácil al usuario el tratamiento de los archivos, como son: el botón *Traslate* (que realiza la conversión del archivo), *Refresh* (que permite volver a leer el archivo desde el disco), *Save* (que graba el archivo en disco) y *Advanced.* Este último botón, se corresponde con la ventana abajo a la derecha mostrada en la imagen anterior. Con *Advanced*, se permite al usuario poder modificar diversas opciones complementarias que tiene la aplicación de forma sencilla e intuitiva.

En esta misma vista se puede realizar una serie de manipulaciones con el menú *Edit.*

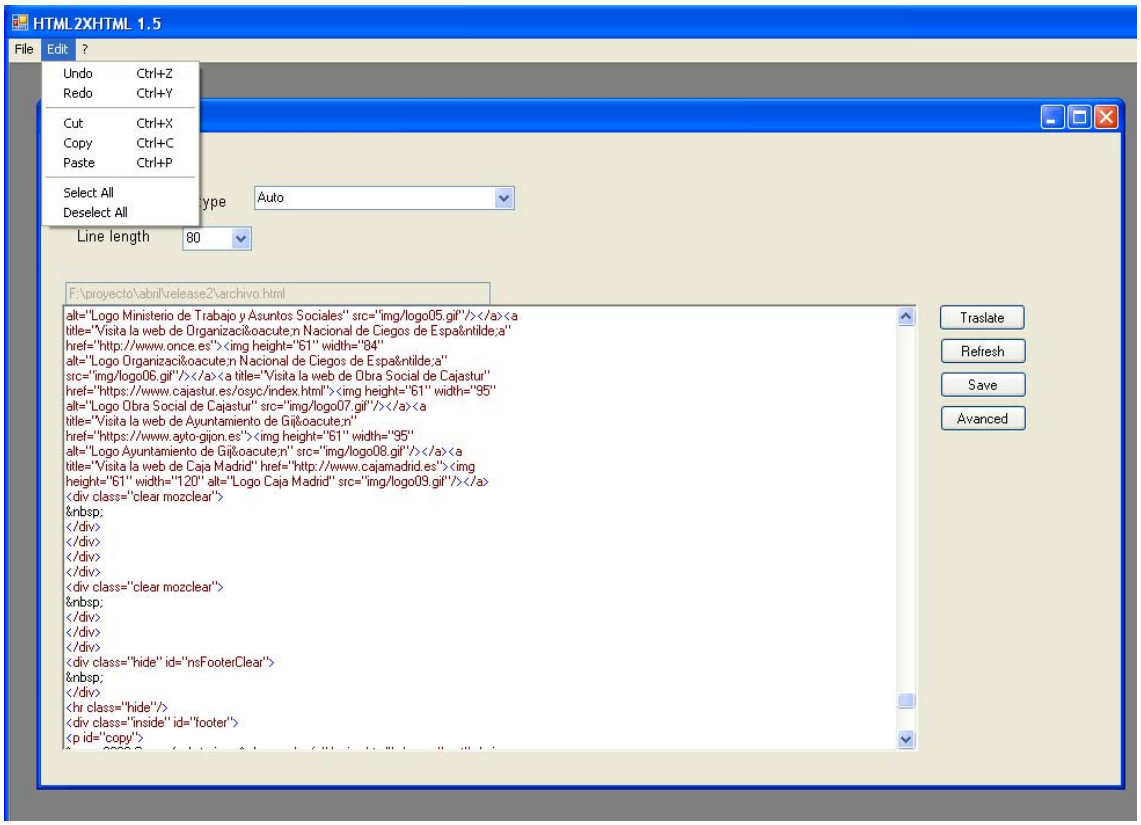



**Figura 8: La Aplicación con el menú *Edit* abierto**

Como se observa en la imagen anterior, el menú *Edit* permite manipular el archivo a convertir de diversas maneras. Así se tiene las opciones *Undo (*Deshacer), *Redo* (Rehacer), *Cut*, *Copy* y *Paste* (Cortar, Copiar y Pegar, respectivamente) y *Select All* y *Deselect All* (Seleccionar Todo, y Deseleccionar Todo).

Otra manera de interactuar con la aplicación es mediante la conversión múltiple de archivos. Para ello, dentro del Menú *File* del programa nos encontramos con la opción *Traslate Multiple Files…*

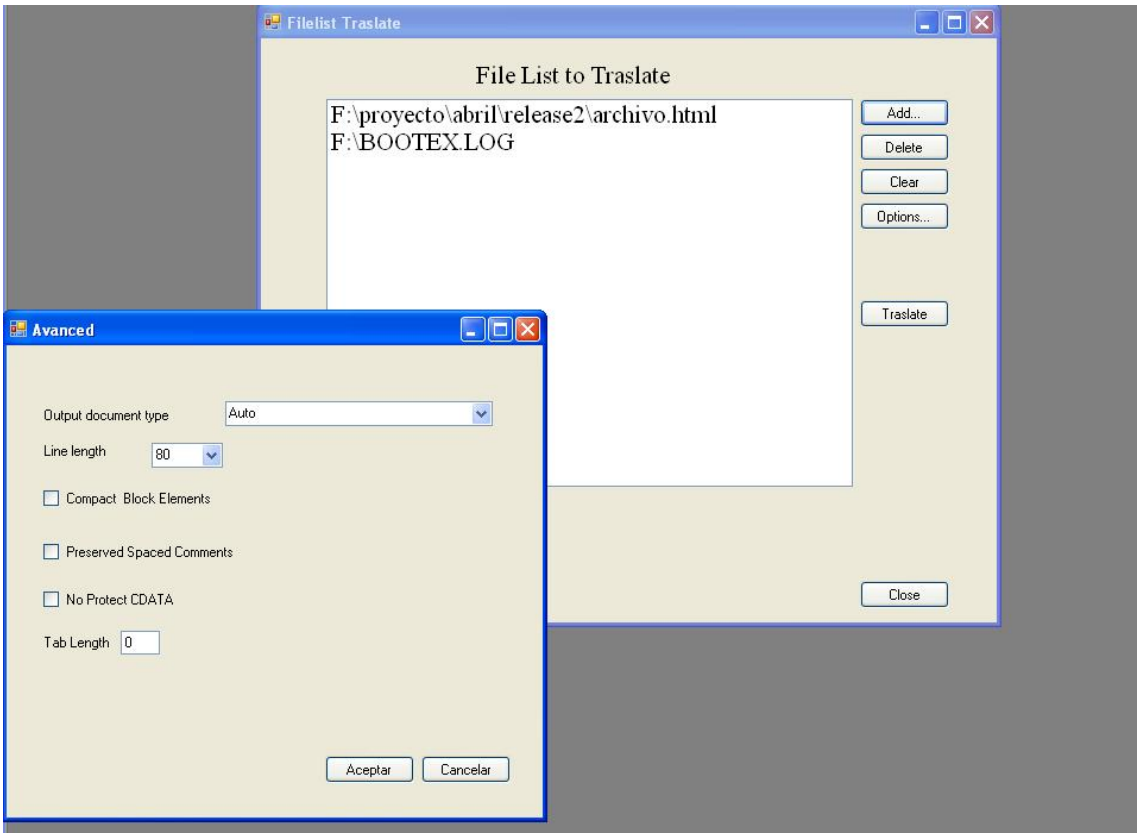



**Figura 9: Ventanas de la conversión múltiple de archivos**

Al hacer clic en la opción, se muestra la vista de la imagen anterior, la cual permite escoger varios archivos diferentes a convertir. Mediante el botón *Add*, se añaden nuevos archivos a la lista y con *Delete*, se elimina el fichero elegido. El botón *Clear* se limpia completamente la lista de los archivos y con la opción *Traslate* comienza la conversión de los archivos. Con *Options…* aparece una segunda ventana (en la imagen, abajo a la derecha), en la que se permite el cambio de las distintas opciones disponibles en la aplicación y que serán comunes a todas las conversiones de la lista de archivos.

# IMPLEMENTACIÓN

En esta parte de la memoria se explicará, de manera pormenorizada, algunas cuestiones importantes sobre la implementación de la aplicación. Se hablará de cómo la aplicación interacciona con el comando ejecutable, cómo funciona el mini editor HTML, de qué manera la aplicación pinta el código fuente y cómo se implementó el menú contextual del editor.

## INTERACCIÓN DEL COMANDO

Como se ha explicado anteriormente, se decidió que la aplicación interaccionara con html2xhtml mediante su interfaz de línea de comandos. Para ello, la aplicación realiza una llamada externa a html2xhtml pasándole los parámetros necesarios para la ejecución. En este apartado se detallará cómo se realiza este paso.

Para usar el comando ejecutable en la aplicación c# hay que realizar unos pasos antes de poder ejecutar el comando:

- Vincular el comando a la aplicación. Para ello se emplea la clase Process de c#, que permite ejecutar un proceso externo a la aplicación. Un componente **Process** proporciona acceso a un proceso que se está ejecutando en un equipo. Un proceso, dicho de un modo sencillo, es una aplicación en ejecución. Un subproceso es la unidad básica a la que el sistema operativo asigna tiempo de procesador. Un subproceso puede ejecutar cualquier parte del código del proceso, incluidas las partes que otro subproceso esté ejecutando en ese momento.

```
1 System.Diagnostics.Process p = new System.Diagnostics.Process();
2 p.StartInfo.FileName = @"html2xhtml.exe";
```

El código anterior establece un enlace entre la aplicación y el proceso (en este caso **htmlxhtml.exe**), para que más adelante pueda ser ejecutado.

- Realizar la cadena de ejecución del comando. Se trata de recoger los distintos parámetros del comando para poder ejecutarlo correctamente dentro de la aplicación.

```
3.  if ((string)comboBox1.SelectedItem=="XHTML 1.0 Transitional ") argumentos=argumentos+"transational";
4. if ((string)comboBox1.SelectedItem == "XHTML 1.0 Frameset ") argumentos = argumentos + "frameset";
```

```
5. if ((string)comboBox1.SelectedItem == "XHTML 1.0 Strict ") argumentos = argumentos + "strict";
6.  if ((string)comboBox1.SelectedItem == "XHTML 1.1 ") argumentos = argumentos + "1.1";
7. if ((string)comboBox1.SelectedItem == "XHTML Basic 1.0 ") argumentos = argumentos + "basic-1.0";
8. if ((string)comboBox1.SelectedItem == "XHTML Basic 1.1 ") argumentos = argumentos + "basic-1.1";
9. if ((string)comboBox1.SelectedItem == "XHTML Print 1.0 ") argumentos = argumentos + "print-1.0";
10.          if ((string)comboBox1.SelectedItem == "XHTML Mobile Profile ") argumentos = argumentos + "mp";
11.          if ((string)comboBox1.SelectedItem == "Auto") argumentos = argumentos + "auto";
12.          argumentos = argumentos + " -l " + comboBox2.SelectedItem.ToString();
13.          if (Compact != null) argumentos = argumentos + " --compact-block-elements";
14.          if (Preserved != null) argumentos = argumentos + " --preserve-space-comments";
15.          if (Data != null) argumentos = argumentos + " --no-protect-cdata";
16.          argumentos = argumentos + " -b " + Tab;
```

Como vemos en el código expuesto, los distintos parámetros del comando se guardan en la variable de texto "argumentos" formando la cadena de ejecución del ejecutable. Cabe destacar la selección del tipo de documento de salida: al ser múltiple se optó por una Lista menú y según la elección del usuario, se guardará en argumentos uno u otro (líneas 1-9).

- Redirigir la salida estándar del comando. Se necesita redirigir la salida estándar del comando ya que no se quiere que se guarde en un archivo en el sistema, sino que se muestre en una caja de texto multilínea para que el usuario pueda ver y comprobar el resultado de la transformación.

```
1 p.StartInfo.RedirectStandardOutput = true;
```

De esta manera tan sencilla se puede redirigir y recoger la salida estándar para poder visualizar el resultado.

- Ejecutar el comando y recoger el resultado.

```
1 p.Start();
2 string output = p.StandardOutput.ReadToEnd();
3 p.WaitForExit();
4 resultado.Text = output;
```

Como se puede observar en el código arriba expuesto, para ejecutar el proceso se utiliza el método *Start* de la clase **process** (línea 1)**,** que inicia la aplicación contenida en la propiedad *Filename*. Para recoger el resultado de la ejecución del proceso se utiliza el método *StandardOutput* y la propiedad *RoadToEnd*. El resultado se guarda en la

variable de texto **output** (línea 2). La propiedad *WaitForExit* (línea 3) es necesaria para que la aplicación espere a la finalización de la recogida de datos para terminar el proceso. Después de esto, se muestra el resultado enviando la salida guardada en **output** a la caja de texto que muestra el código (línea 4).

# MINI-EDITOR DE HTML

Una de las funcionalidades de esta aplicación es poder modificar el archivo de HTML abierto. Para ello es necesario que el usuario pueda cambiar el contenido del texto. Para ello se ha optado por tener un objeto **RichTextBox** multilínea. Este control, es similar a un TextBox pero que posee más opciones. A diferencia del control TextBox clásico, este permite por ejemplo: cambiar el tipo de fuente, mostrar imágenes, cargar archivos de texto enriquecido (archivos RTF ), leer y guardar archivos mediante una serie de métodos que posee el control, tanto RTF como TXT, buscar cadenas de texto dentro del control con un método propio, y muchas otras opciones.

```
1 this.resultado = new System.Windows.Forms.RichTextBox();
```

De este modo se inicia un objeto de la clase **RichTextBox** para poder luego utilizarlo en la aplicación. El objeto permite, por sí solo, la edición de texto dentro de él, por lo que se necesita crear un menú contextual y asegurar que las modificaciones que hagamos antes de ejecutar el comando sean pasadas a éste.

Para crear el menú contextual es necesario escribir el siguiente fragmento de código:

```
1 this.contextMenuStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
i.   this.toolStripMenuItem1,
ii.  this.toolStripSeparator2,
iii. this.copyToolStripMenuItem,
iv.  this.pasteToolStripMenuItem,
v.   this.cutToolStripMenuItem,
vi.  this.toolStripSeparator1,
vii. this.selectAllToolStripMenuItem,
viii. this.deselectAllToolStripMenuItem});
2 this.contextMenuStrip1.Name = "contextMenuStrip1";
3 this.contextMenuStrip1.Size = new System.Drawing.Size(153, 148);
```

Con este código podemos inicializar un objeto **contextMenu** en el que, en este caso, se han añadido las opciones de *copiar*, *cortar*, *pegar*, *seleccionar todo* y *deseleccionar todo*. Además, y por comodidad, se añadió la opción de *Traducir*.

Para hacer funcionar el menú hay que inicializar los eventos y funciones correspondientes (ver código al final del proyecto) .

Además del menú contextual, se debe asegurar que los cambios introducidos en el documento Html deben ser enviados directamente al comando ejecutable. Para ello, y en ese momento en la persona usuaria pulse el botón de traducir, se guardar en un fichero temporal el contenido de la caja de texto para que el comando pueda traducirlo.

```
1 saveFileDialog1.FileName =directorio+"\\temporal2.html";
2 string input = resultado.Text;
3 using (Stream stream = saveFileDialog1.OpenFile())
4 {

5 using (StreamWriter writer = new StreamWriter(stream))
{
6     writer.Write(input);

7 }
8 }
9 String argumentos =  directorio+"\\temporal2.html"+ " -t ";
```

Como se puede ver en el código anterior utilizamos un objeto *saveFileDialog* para guardar el Html. Para ello se introdice el código en una variable de texto(2), y uilizamos un stream(en realidad es un buffer) y un tipo *writer*, que nos escribe el contenido de la variable de texto *input* en el fichero(6). Por lo tanto, antes de llamar al comando ejecutable.

# Coloreado del código fuente

Una de las características fundamentales de un buen editor de texto es poder identificar fácilmente las partes de ese código. Para ello se colorean las etiquetas (en

inglés *tags*) que nos ayudan a poder situarnos dentro de él. En este caso se ha optado por colorear las etiquetas de un solo color ya que al ser solamente HTML, no parece necesario la utilización de otros colores adicionales. Para ello se ha creado una función que permite colorear las etiquetas del texto. Esto es posible ya que el control **RichTextBox** permite colorear código ya que se puede utilizar el texto en formato RTF.

El formato RTF (siglas en inglés para *Rich Text Format* o 'Formato de texto enriquecido') surgió como acuerdo para intercambio de datos entre Microsoft y Apple en los tiempos en que Apple dominaba el mercado de los computadores personales. Las primeras versiones del formato .doc de Word derivaban del RTF. Incluso ahora hay programas de Microsoft, tal como Wordpad, que usan directamente RTF como formato nativo. El documento en formato RTF tiene extensión .rtf. Es un formato de texto compatible, en el sentido que puede ser migrado desde y hacia cualquier versión de Word, e incluso muchos otros procesadores de textos y de aplicaciones programadas. El RTF es una forma particular para dar formato a un texto, salvando las diferencias, como lo puede ser HTML o Tex, insertando códigos particulares entre el texto.

Gracias a esta peculiaridad, se puede recorrer para poder introducir el formato necesario al texto(en este caso el color rojo). Además, se pintará de verde los comentarios y de azul el resto del texto.

Como se puede observar en el código(mostrado completamente al final de este proyecto) lo primero que se realiza es pasar el texto a rtf:

```
1 string strRTF = resultado.Rtf;
```

Después se crea una tabla donde van a ir los colores que se van a necesitar, si esa tabla no existe se crea, si está, se borrar para crear una nueva. Esta tabla se situará en la cabecera del código en formato RTF .

Más tarde, ya se puede analizar el código(siempre en RTF), e introduciendo las etiquetas de color donde se necesiten, para finalmente, guardar el texto en el objeto **Richtexbox** para mostrárselo al usuario(línea 27).

```
1 for (int i = 0; i < strRTF.Length; i++)
2 {
3 if (strRTF[i] == '<')
4 {
5 if (strRTF[i + 1] == '!')
6 strRTF = strRTF.Insert(i + 4, "\\cf2 ");
7 else
8 strRTF = strRTF.Insert(i + 1, "\\cf1 ");
9 strRTF = strRTF.Insert(i, "\\cf3 ");
10          i += 6;
11          }
12          else if (strRTF[i] == '>')
13          {
14          strRTF = strRTF.Insert(i + 1, "\\cf0 ");
15          if (strRTF[i - 1] == '-')
16          {
17          strRTF = strRTF.Insert(i - 2, "\\cf3 ");
18          i += 8;
19          }
20          else
21          {
22          strRTF = strRTF.Insert(i, "\\cf3 ");
23          i += 6;
24          }
25          }
26          }
27          resultado.Rtf = strRTF;
28          }
```

# MENÚ CONTEXTUAL

El menú contextual de la aplicación se va a utilizar para que el usuario pueda interactuar con la aplicación más fácilmente sin tener que utilizar el menú de edición del programa. En Visual C# lo único que tenemos que realizar es arrastrar un item de menú hacia nuestra aplicación, y comenzar ha hacer las funciones de copiar, cortar , pegar, etc en nuestro código. A continuación se muestra la forma de realizar el menú contextual nombrado aquí como cutToolStripMenuItem.

```
this.toolStripMenuItem1.Name = "toolStripMenuItem1";
```

```
this.toolStripMenuItem1.Size = new System.Drawing.Size(152, 22);
this.toolStripMenuItem1.Text = "Traslate Now...";
this.toolStripMenuItem1.Click += new System.EventHandler(this.button3_Click);
//
// toolStripSeparator2
//
this.toolStripSeparator2.Name = "toolStripSeparator2";
this.toolStripSeparator2.Size = new System.Drawing.Size(149, 6);
//
// copyToolStripMenuItem
//
this.copyToolStripMenuItem.Name = "copyToolStripMenuItem";
this.copyToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
this.copyToolStripMenuItem.Text = "Copy";
this.copyToolStripMenuItem.Click += new System.EventHandler(this.copyToolStripMenuItem_Click);
//
// pasteToolStripMenuItem
//
this.pasteToolStripMenuItem.Name = "pasteToolStripMenuItem";
this.pasteToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
this.pasteToolStripMenuItem.Text = "Paste";
this.pasteToolStripMenuItem.Click += new System.EventHandler(this.pasteToolStripMenuItem_Click);
//
// cutToolStripMenuItem
//
this.cutToolStripMenuItem.Name = "cutToolStripMenuItem";
this.cutToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
this.cutToolStripMenuItem.Text = "Cut";
this.cutToolStripMenuItem.Click += new System.EventHandler(this.cutToolStripMenuItem_Click);
//
// toolStripSeparator1
//
this.toolStripSeparator1.Name = "toolStripSeparator1";
this.toolStripSeparator1.Size = new System.Drawing.Size(149, 6);
//
// selectAllToolStripMenuItem
//
this.selectAllToolStripMenuItem.Name = "selectAllToolStripMenuItem";
this.selectAllToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
this.selectAllToolStripMenuItem.Text = "Select All";
this.selectAllToolStripMenuItem.Click += new System.EventHandler(this.selectAllToolStripMenuItem_Click);
//
// deselectAllToolStripMenuItem
//
this.deselectAllToolStripMenuItem.Name = "deselectAllToolStripMenuItem";
```

```
this.deselectAllToolStripMenuItem.Size = new System.Drawing.Size(152, 22);
this.deselectAllToolStripMenuItem.Text = "Deselect All";
this.deselectAllToolStripMenuItem.Click += new System.EventHandler(this.deselectAllToolStripMenuItem_Click);
```

# COMPATIBILIDAD ENTRE ENTORNOS .NET Y MONO

Aunque la mayoría de la aplicación es compatible entre los dos entornos y en los dos sistemas operativos, tanto Windows como Linux, sí que hay que hacer algún cambio reseñable para que la aplicación funcione correctamente en ambos sistemas.

El cambio reseñado es referente a la utilización del *richtextbox* en Linux. Al tener que utilizar en Linux bibliotecas propias de Mono, las funciones de coloreado de este elemento aún no están disponibles, por lo que en la versión disponible para Linux, estas características no son visibles. Aún así, se ha decidido dejar en el código fuente de esta versión las funciones de coloreado ya que es posible que en un futuro puedan estar disponibles en las siguientes versiones de Mono.

# PRUEBAS

Todo proyecto software recién realizado necesita, antes de su puesta a disposición del cliente, una serie de pruebas para comprobar su correcto funcionamiento. Esta batería de pruebas se diseña casi siempre de forma automática para facilitar su ejecución cubriendo diversos casos de prueba.

La prueba no es una actividad sencilla, no es una etapa del proyecto en la cual se asegura la calidad, sino que la prueba debe ocurrir durante todo el ciclo de vida: podemos probar la funcionalidad de los primeros prototipos; probar la estabilidad, cobertura y rendimiento de la arquitectura; probar el producto final, etc. Lo que conduce al principal beneficio de la prueba: tener tiempo de reaccionar mientras hay todavía recursos para hacer algo.

Como este proyecto es una interfaz gráfica es más difícil realizar pruebas automáticas. En este caso se realizarán pruebas más informales en las que se interactuará con la aplicación de forma manual. Los casos de prueba se van a dividir en tres apartados:

- Pruebas generales del sistema y casos conflictivos
- Pruebas de tamaño de archivos
- Pruebas de funcionamiento en Windows y Linux

## PRUEBAS GENERALES Y CASOS CONFICTIVOS

En esta parte del capítulo, se verán las pruebas generales y se mostrarán algunos casos conflictivos para comprobar si la aplicación los resuelve correctamente o si existe algún fallo subsanable. Se comprobará con archivos con tildes (para comprobar signos de puntuación) y también con rutas de archivos en los cuales haya un espacio en blanco o más para ver como la aplicación trata ese archivo.

- Archivo con tildes. Hemos elegido un archivo mediano de 22000 bytes cuyo nombre es archivó.html.

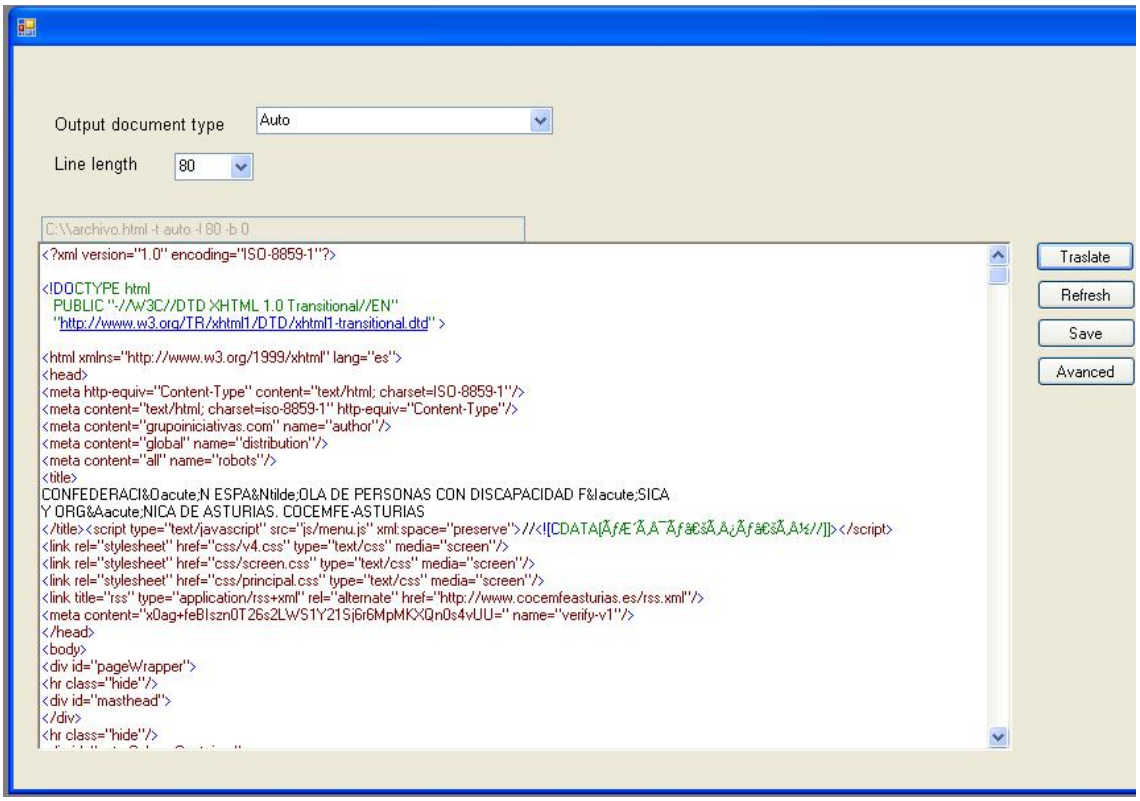



**Figura 10: Salida de traducción de archivo con tildes**

Como se puede observar, a la aplicación no le afectan los archivos con signos de puntuación. El proceso de traducción de este tipo de archivos se hace correctamente independientemente de si tiene signos de puntuación o no. También se probó con diéresis y guiones con idéntico resultado.

- Archivos con directorios con espacios en banco. Se ha escogido un archivo de 21000 bytes que está en el directorio **C:\Documents and Settings.**

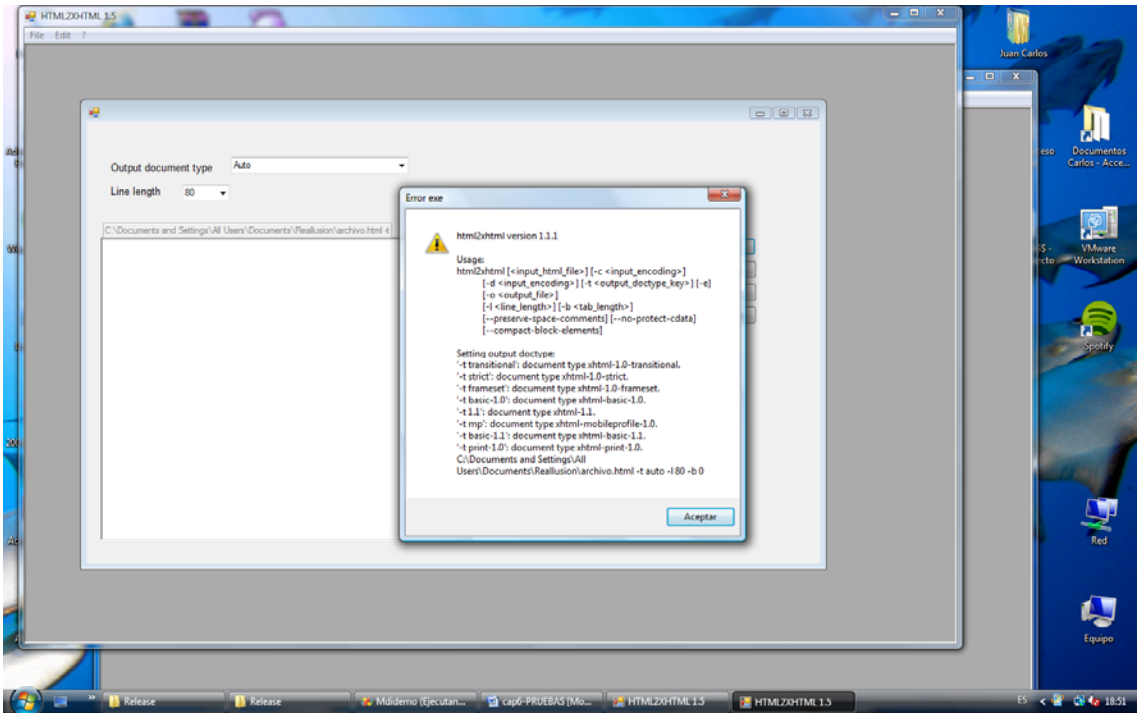



**Figura 11:Salida error de traducción**

Como se puede observar, al realizar la traducción se ha producido un error del comando html2xhtml. Investigando, el error se produce en la forma de mandar los argumentos del archivo a traducir, concretamente en la manera de mandar la dirección de los archivos. Al ser la ejecución de html2xhtml en entorno DOS, no existen nombres de directorio de más de 8 caracteres. Por eso, en las nuevas versiones de Windows, se realiza un atrajo, añadiendo “”” al comienzo y al final de la dirección del fichero.

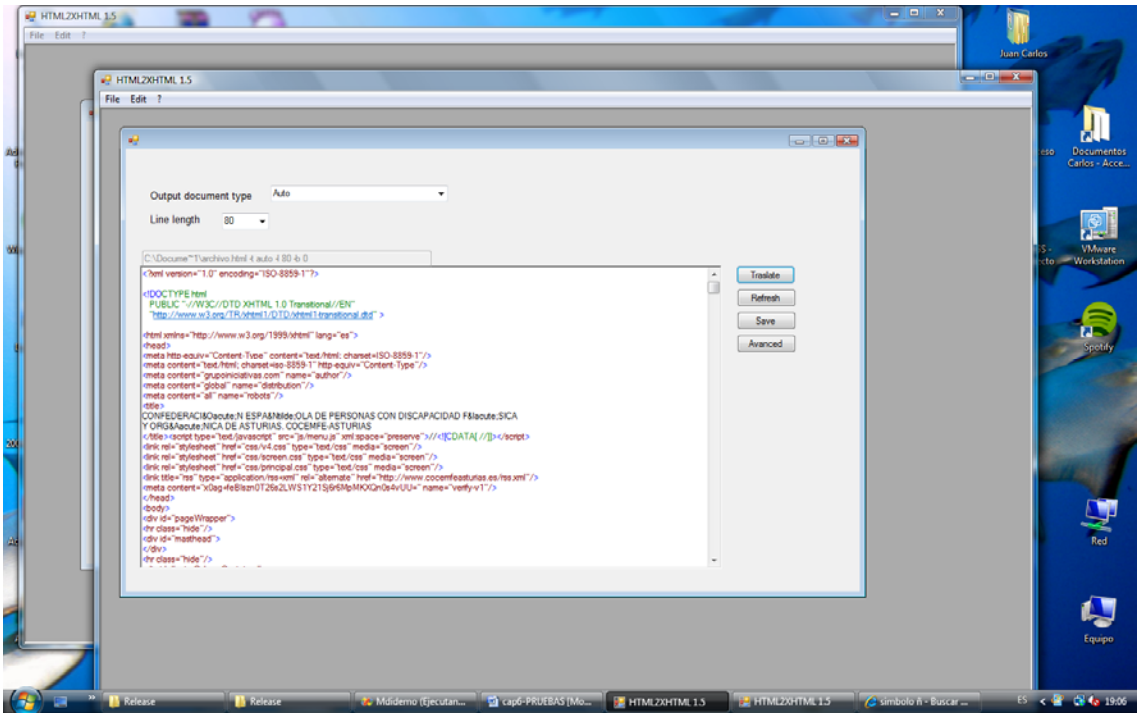



**Figura 12: Salida archivo con espacios**

Con ello, se comprueba que la traducción es correcta.

# PRUEBAS DE TAMAÑOS

En este caso se escogerán archivos de formato HTML de distintos tamaños para comprobar que la aplicación los soporta. Concretamente se seleccionaran 3 archivos:

- El primero será de una página Web, de 22000 bytes cuyo contenido es exclusivamente HTML (HTML 1.0 Transitional).

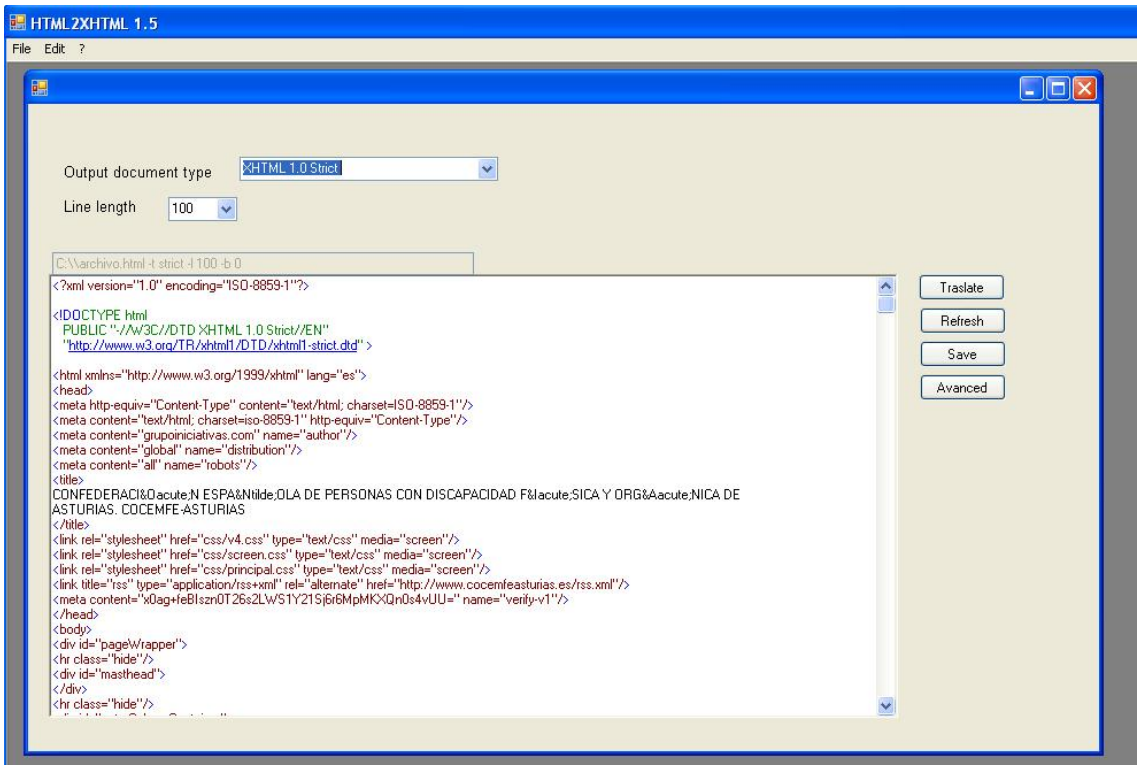



**Figura 13:Salida traducción archivo tamaño medio**

Como se observa, se ha traducido este archivo a XHTML 1.0 Strict sin ninguna complicación. Además se puso un tamaño de línea de 100 para hacer la llamada a la aplicación DOS más complicada. A pesar de ello, como se ve en la imagen superior, la traducción se hizo automáticamente y sin problemas.

- El segundo de los archivos a probar corresponde a otra página Web, en este caso cerca de 45000 bytes y también en HTML 1.0 Transitional

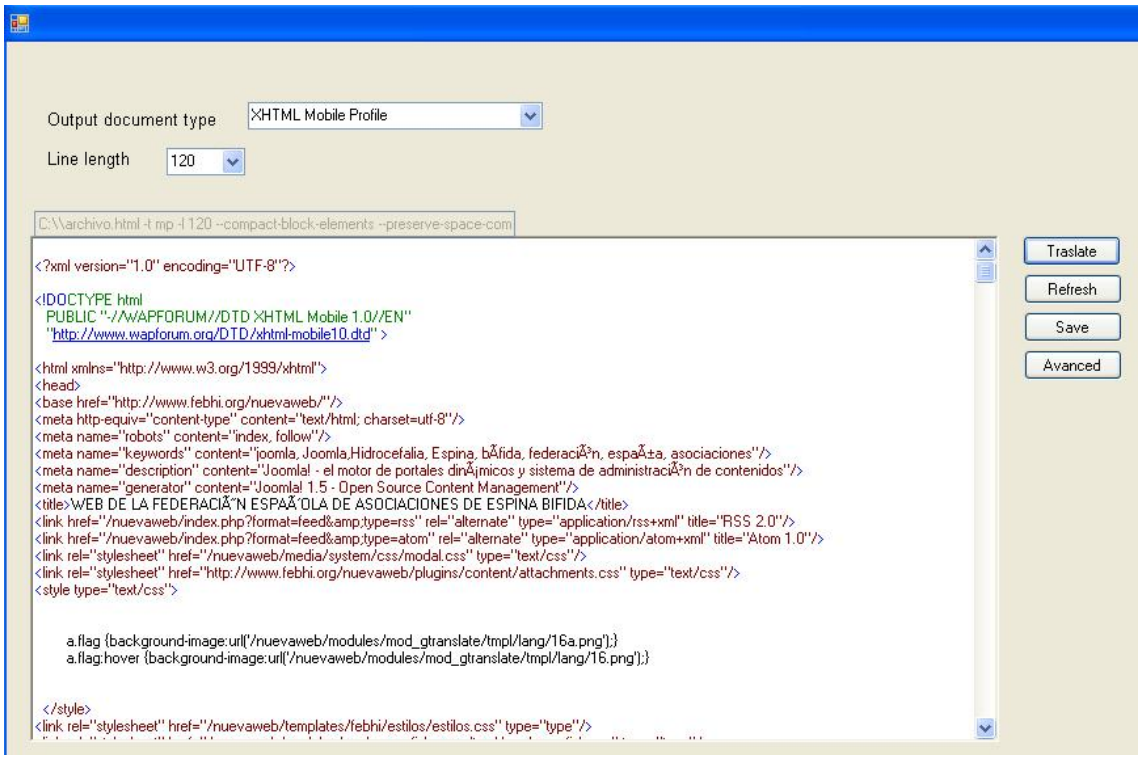



**Figura 14: Salida traducción archivo grande**

Como se ve en la imagen adjunta, se ha decidido traducir el archivo a XHTML Mobile profile. Además se dispuso activar todas las opciones disponibles en la sección de Avanzadas para ver como se comporta la aplicación y como se ve en la imagen superior, la traducción se hace correctamente, sin ningún tipo de error.

- El último de los casos de prueba será un pequeño fichero realizado para la ocasión que solo tiene las etiquetas básicas de cualquier archivo HTML:

```
<head>

<title> Prueba</title>

</head>

<body> Esta es una pagina de Prueba</body>

</html>
```

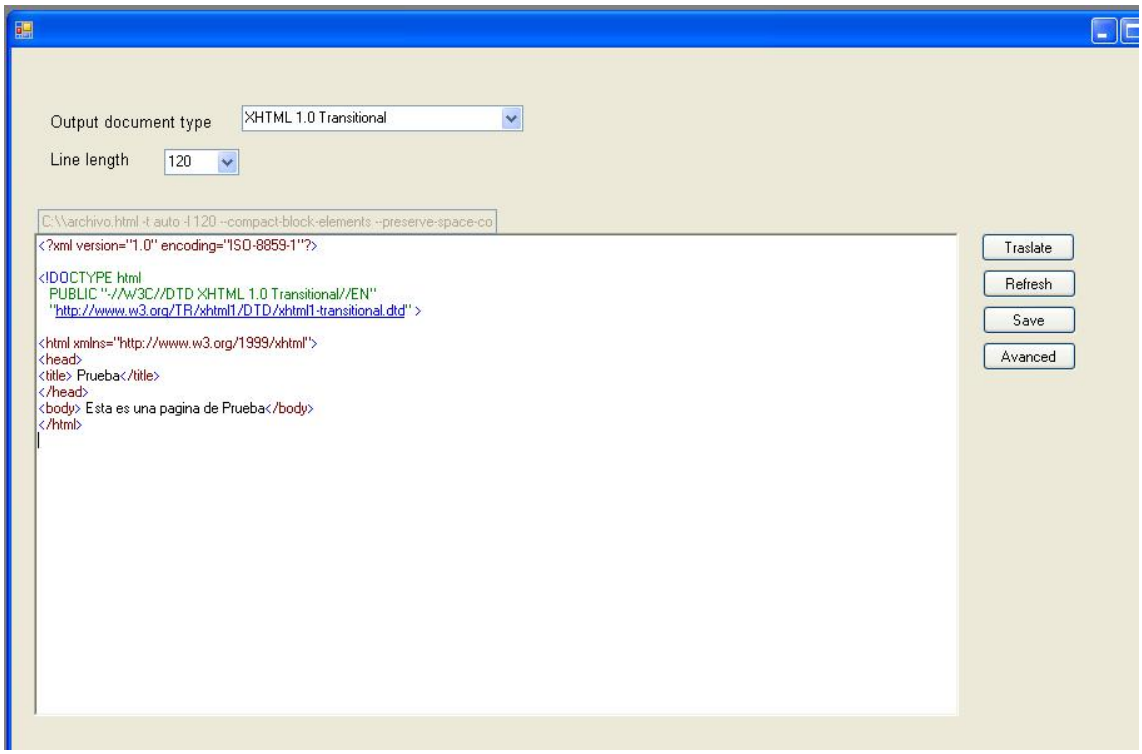



**Figura 15: Salida traducción archivo pequeño**

Como se aprecia en la imagen adjunta, al traducir este archivo simple, no existe ningún error. Además, este archivo, como el anterior, se tradujo con las opciones avanzadas activadas.

- La última prueba de este apartado la forman los tres archivos en conjunto para probar la traducción simultánea de varios archivos.

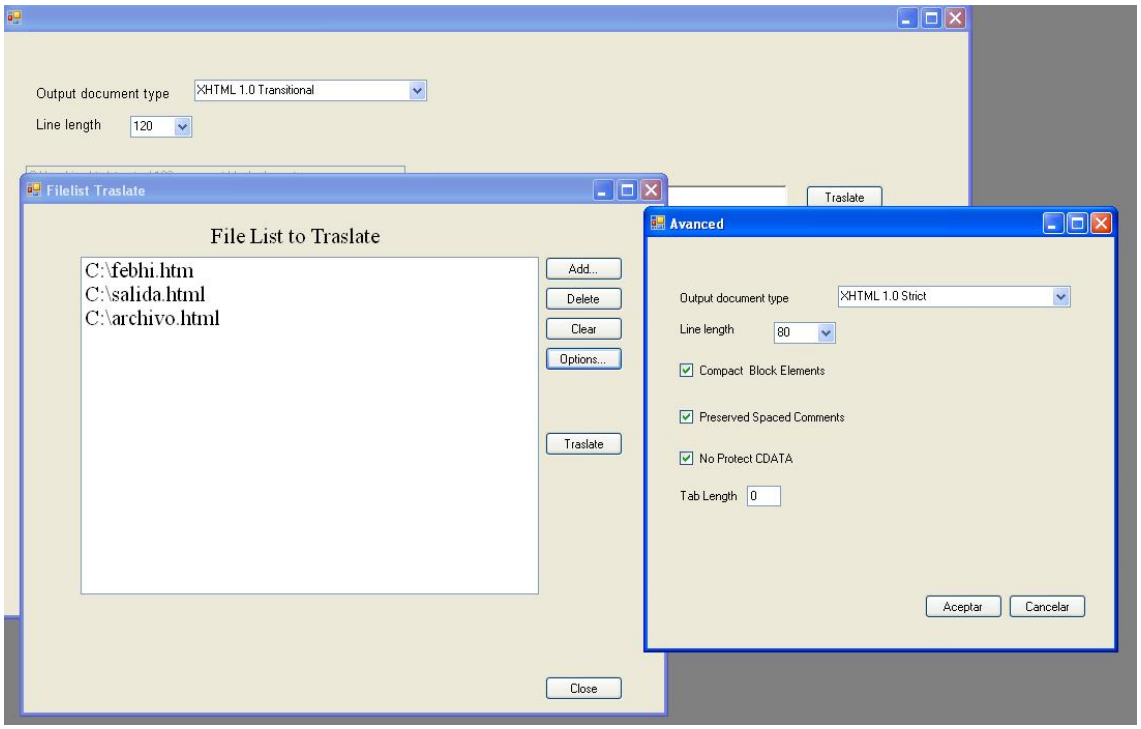



**Figura 16:Salida traducción múltiple**

Como muestra la imagen, se traducen los archivos con todas las opciones disponibles y al XHTML 1.0 Transitional. Comprobando la salida se observa que todos y cada uno de los ficheros que han sido traducidos lo han hecho de forma correcta.

# PRUEBAS DE FUNCIONAMIENTO EN WINDOWS-LINUX

Estos casos de prueba se realizarán para comprobar el funcionamiento tanto en Windows (con Mono 2.0) y Linux. En Windows ya se ha realizado pruebas sin Mono (en el apartado anterior) por lo que en este caso se realizará solo con el paquete Mono 2.0. Además en este apartado se realizarán las pruebas de la conversión múltiple de archivos.

Para ello, se van a utilizar los mismos archivos que para las pruebas anteriores y así poder comprobar si hay alguna diferencia sustancial con respecto al programa de pruebas efectuado en Windows sin Mono.

• **Pruebas en Windows y Mono 2.0.** Para ello hay que iniciar la aplicación con el paquete Mono desde Conmmand (CMD) con la siguiente sentencia:

*Mono Convertxhtml.exe*

Se escogerá un archivo para la prueba de los anteriormente utilizados. Específicamente, se utilizara el archivo de la página web de Cocemfe Asturias.

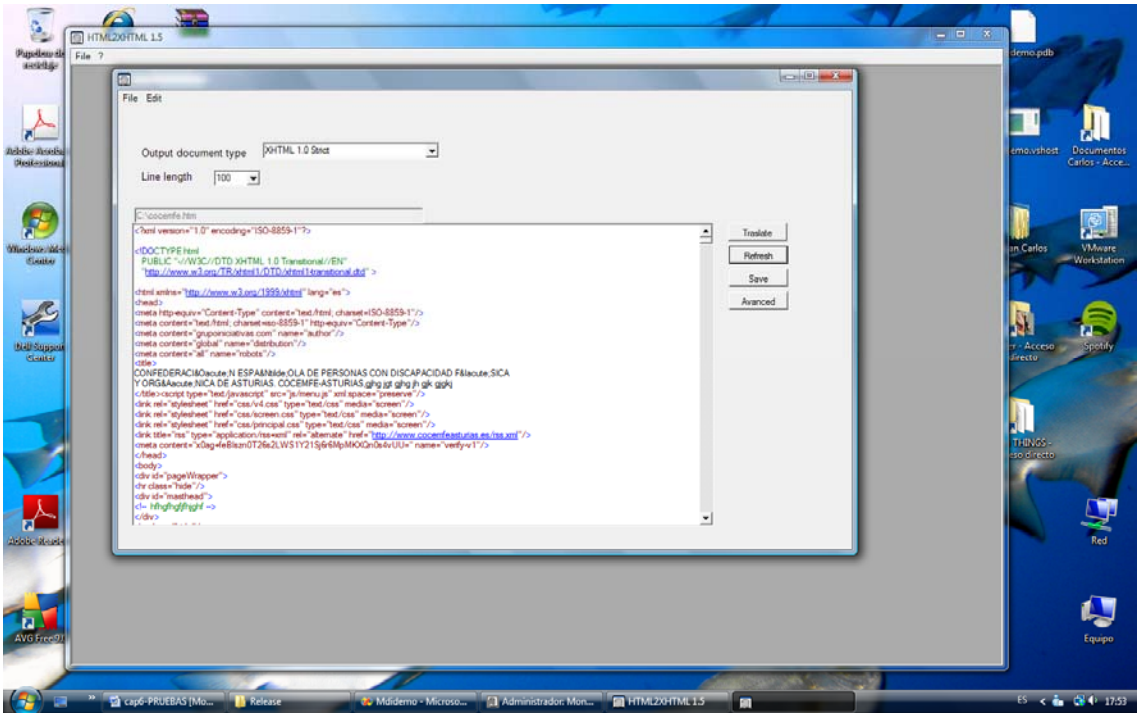



**Figura 17: Salida archivo aplicación con Mono 2.0**

Como se puede observar, la aplicación, cuando se ejecuta en entorno Mono, tiene un aspecto ligeramente diferente que cuando se ejecuta en entorno .NET. Aún así, el programa se ha ejecutado y ha traducido el archivo correctamente. Cabe destacar que en esta ejecución, y al abrir el archivo, se produce un retraso de unos segundos al mostrar la siguiente vista.

- **Pruebas en Linux**

Para Linux se utilizará una distribución Ubuntu en la que se instalará el paquete Mono 2.0 con Monodevelop y la aplicación html2xhtml en su versión Linux. Se probará con el archivo de la página web de 22000 bytes.

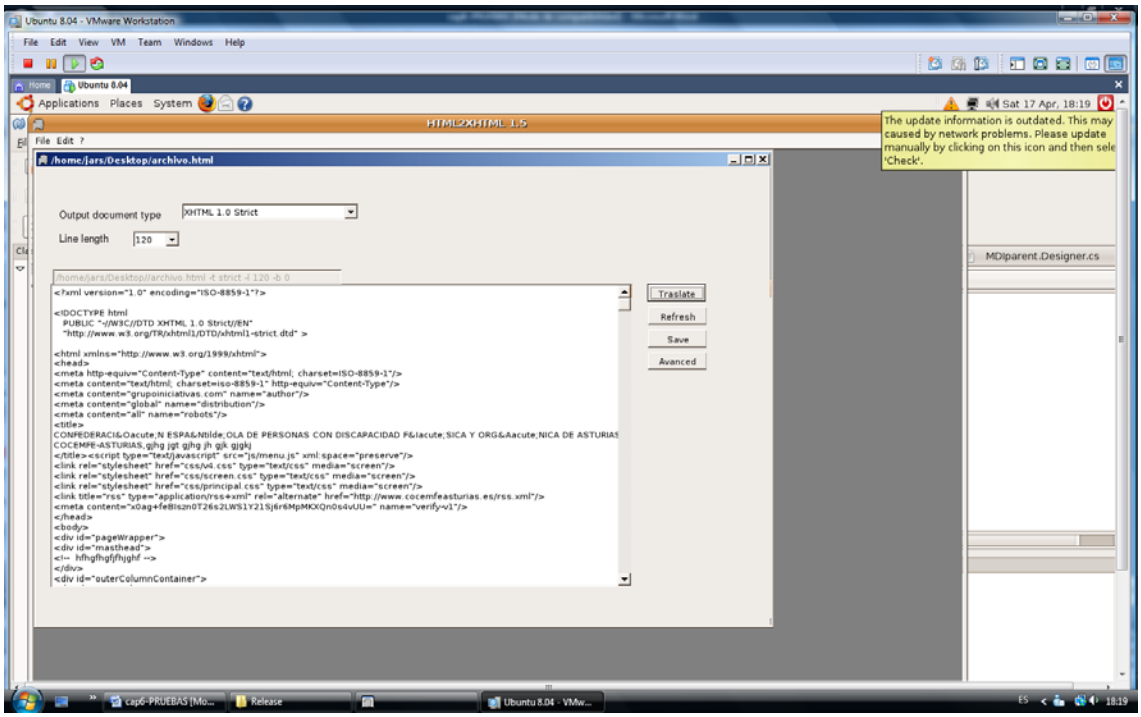



**Figura 18: Salida archivo en Linux**

Se observa que la aplicación realiza correctamente la traducción del fichero sin ningún tipo de problemas.

A continuación se probará la traducción simultánea de varios archivos en Linux.

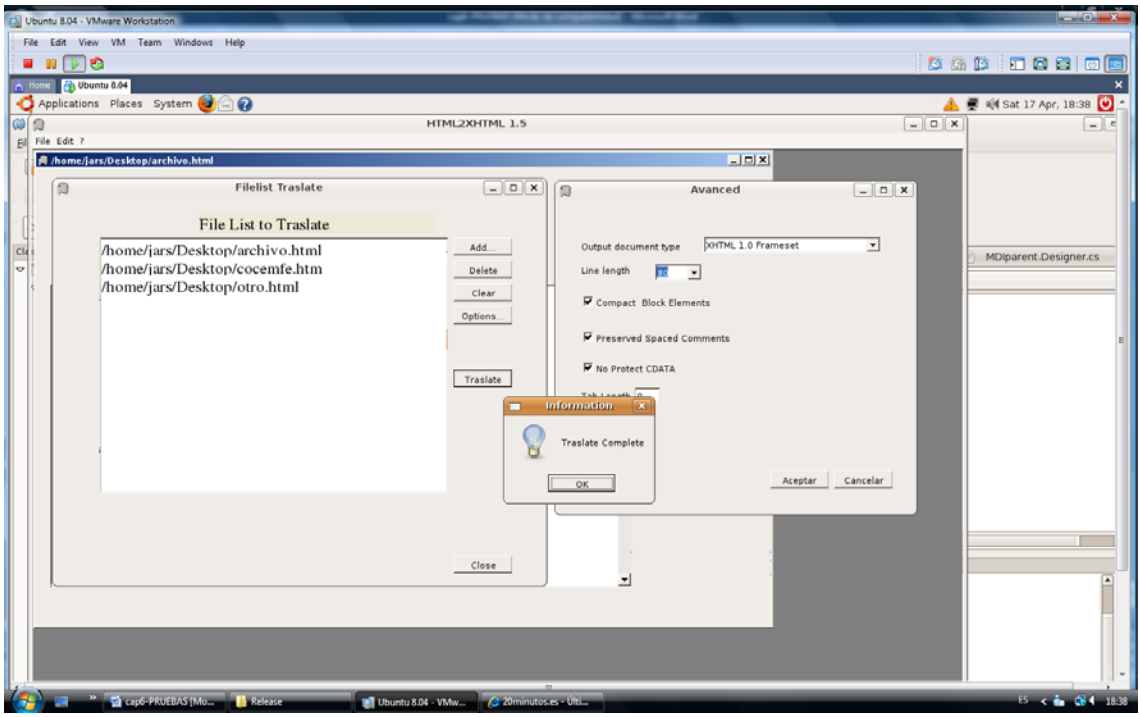



**Figura 19:Salida traducción archivos múltiples en Linux**

Como se observa en la imagen superior, y después de verificar los archivos resultantes, la traducción de los tres archivos ha sido correcta y sin ningún tipo de incidencia.

# CONCLUSIONES Y PLAN DE FUTURO

Como se ha podido comprobar en este proyecto fin de carrera, la aplicación desarrollada puede funcionar tanto en Windows como en Linux de forma análoga gracias al entorno de ejecución proporcionado por el proyecto Mono.

El mayor reto de este proyecto consistió, en principio, en buscar información sobre el entorno Mono, en su mayoría escasa. Poco a poco, y coincidiendo con la realización de la aplicación fueron surgiendo distintos foros y páginas relativas a Mono, con lo que el problema de la información quedaba subsanado.

En la realización del proyecto surgieron dudas sobre su funcionamiento general ya que, al fin y a cabo, Mono es una *emulación* de parte del entorno .NET de Microsoft. Por ese motivo, algunas de las instrucciones utilizadas para nuestra aplicación o no funcionaban o el resultado no era el esperado.

El desarrollo del proyecto Mono no está muy avanzado. Sin embargo el estudio a fondo de este paquete permite vislumbrar un futuro prometedor para las aplicaciones desarrolladas sobre el entorno .NET dependiente de Microsoft. Hace años nadie se imaginaba poder ejecutar una aplicación específica para Windows en cualquier ordenador equipado con Linux, a menos que éste emulara un Windows virtual. Aun así, los programas no terminaban de ejecutarse a la perfección porque en realidad no era un verdadero Windows.

Con Mono no solo se pueden ejecutar los programas escritos en .NET, sino que también es posible, con su emulación de c# (llamada Gtk#) ejecutar en Windows aplicaciones escritas y/o compiladas en Linux mediante Gtk#.

Un trabajo futuro a partir de este proyecto podría consistir en investigar y realizar una aplicación en el sentido contrario, es decir, realizar el código en Mono y averiguar

si ejecutándolo en Windows con entorno .NET lo hace de forma correcta. Para ello habría que estudiar cuidadosamente la alternativa a C# que da Mono: Gtk#.

Tras la realización de este proyecto resulta interesante seguir investigando sobre los posibles usos de aplicaciones de .NET en sistemas Linux puesto que, como se ha visto, con el Software Libre Mono 2.0 se puede compilar y ejecutar una aplicación realizada en Visual C# en cualquier sistema Unix con resultados satisfactorios.

En este proyecto se ha realizado una profunda investigación sobre las tecnologías necesarias para poder realizar una GUI del programa html2xhtml compatible con los sistemas Linux. Se ha decidido utilizar el comando ejecutable ya que parece la forma más cómoda de realizar la interfaz gráfica de modo que ésta, haga las llamadas a html2xhtml directamente. Pero este trabajo no debería quedar aquí.

Para futuros proyectos se recomienda que se opte por implementar una biblioteca DLL e incluirla en la interfaz. De esta manera se asegura la total compatibilidad de la aplicación al no depender de llamadas a funciones de programas externos ajenos.

Además, también se recomienda en futuros proyectos, investigar la posibilidad de realizar la aplicación en un lenguaje nativo de Mono 2.0, en este caso se optaría por Gtk#. El lenguaje Gtk# es también compatible con Windows si se ejecuta bajo el entorno Mono y, aunque su desarrollo aun está retrasado, valdría la pena realizar un proyecto fin de carrera para la investigación y desarrollo de este lenguaje y como colofón, realizar una aplicación para este entorno y comprobar su compatibilidad con Windows, puesto en Linux se presupone.

# PRESUPUESTO

En esta parte de la memoria se cuantificará el esfuerzo necesario para sacar adelante el proyecto con los objetivos definidos en la parte inicial. A continuación se expondrán las horas necesarias para la realización de las tareas encaminadas a la realización de este proyecto, proponiendo una tasa en Euros del total de este trabajo. Suponemos que se trabaja con una persona a jornada completa (8h, 20 dias/mes), cuyo salario hora es de 20€.

| Tarea | Días | Coste/día | Coste Total |
|---|---|---|---|
| **Estudio del problema** | **6** | **160 €** | **960,00 €** |
| **Estudio de Tecnologías Web** | **12** | **160 €** | **1920,00 €** |
| **Estudio de requerimientos** | **5** | **160 €** | **800,00 €** |
| **Realización Vistas de la Aplicación** | **23** | **160 €** | **3680,00 €** |
| **Recoger variables de la aplicación** | **6** | **160 €** | **960,00 €** |
| **Realización de la ejecución de la cadena del programa** | **7** | **160 €** | **1120,00 €** |
| **Desarrollo de la conversión múltiple de archivos** | **17** | **160 €** | **2720,00 €** |
| **Desarrollo de** | **6** | **160 €** | **960,00 €** |

| | | | |
|---|---|---|---|
| **pruebas** | | | |
| **Desarrollo Documentación** | **41** | **160 €** | **6560,00 €** |

Como se puede ver el coste total del salario para una persona es de **19680 €,** sin contar el espacio necesario para llevarlo a cabo y el ordenador con los programas necesarios.

# MANUAL DE USUARIO

Este apartado detalla el manejo de las distintas opciones (funcionalidades del sistema) que ofrece la aplicación, con la finalidad de que el usuario pueda extraer el máximo rendimiento posible de ésta. Este manual está dividido en las principales funcionalidades y por cada una de ellas se verá su correspondiente descripción funcional, operaciones posibles a realizar, así como los posibles errores y causas.

## INICIAR LA APLICACIÓN

Para iniciar la aplicación hay que distinguir dos casos:

- Usuarios Windows: Para iniciar la aplicación en Windows, simplemente hay que hacer doble clic en el archivo Convertxhtml.exe
- Usuarios Linux: Para iniciar el programa en una distribución Linux, hay que abrir un Shell y teclear “Mono Converxhtml.exe”.

## ABRIR ARCHIVOS

Permite llevar a cabo la inclusión de ficheros a modo de ventanas en la aplicación para su posterior traducción. Se puede ejecutar de dos maneras

- Al inicio de la aplicación, se puede optar por hacer clic en el botón *Open* situado en la vista inicial de saludo,
- O desde el menú *File→Open New Tab…* (Figura 1).

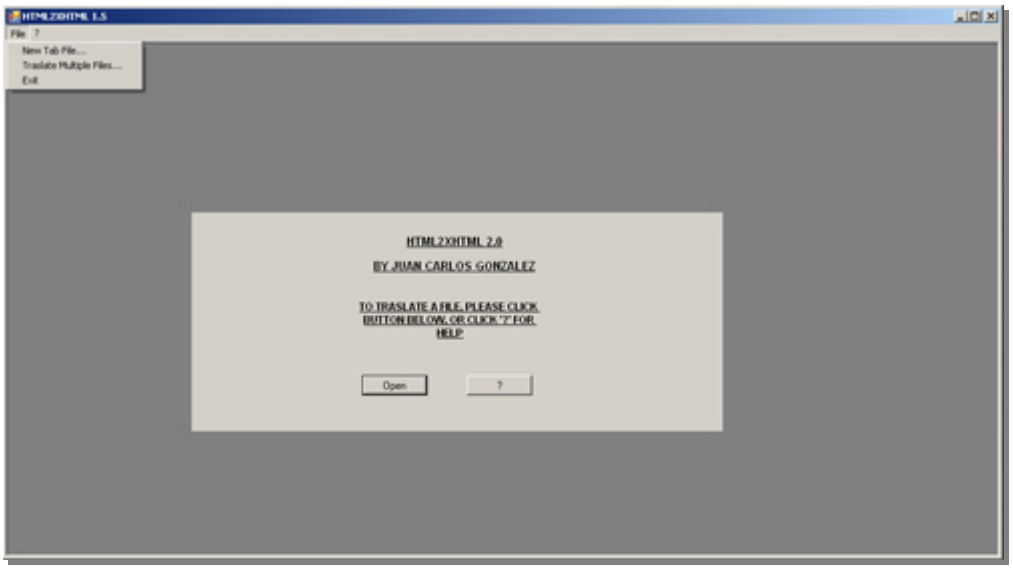



**Figura 20:Vista inicial de html2xhtml**

Al hacer clic en cualquiera de las dos opciones aparecerá un cuadro de diálogo (Figura 2) que permitirá elegir los archivos de formato html para mostrarlos. Recordar que el programa está diseñado para poder elegir varios archivos a la vez, por lo que se podrá elegir uno o varios archivos simultáneamente.

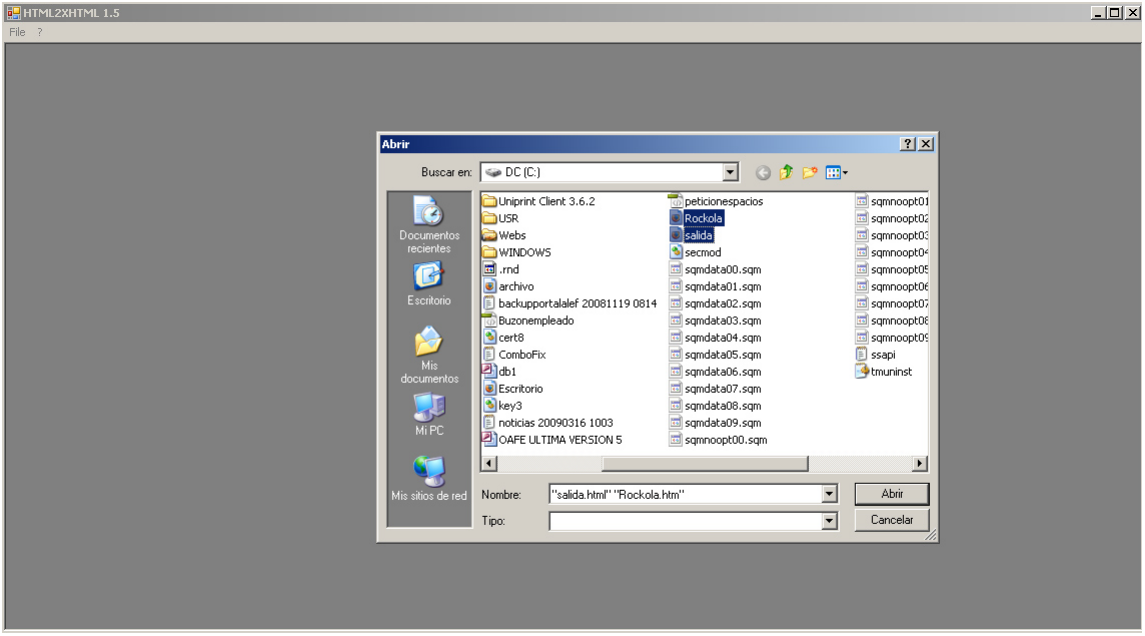



**Figura 21: Cuadro de diálogo para abrir ficheros**

# OPCIONES DE LA APLICACIÓN

En el momento de abrir los archivos, éstos estarán en ventanas independientes, en las cuales, se podrá modificar el texto en cualquier momento, antes y/o después de la transformación. Además se podrá copiar, cortar, pegar, seleccionar o deseleccionar todo el texto y traducir rápidamente con el menú contextual al que se accede haciendo clic con el botón derecho dentro de la ventana que nos muestra el código del archivo (Figura 3), o también seleccionando Edit de la barra de menús.

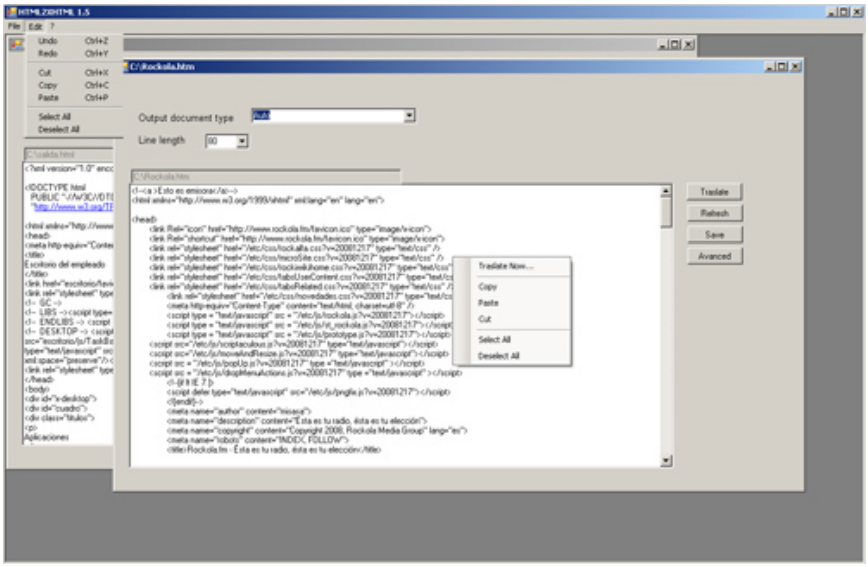



**Figura 22: Ventanas de documentos con el menú contextual**

En cada una de las ventas de los documentos existen dos opciones que son obligatorias para que el programa pueda traducir correctamente los archivos (Figura 4):

- *Output document type*: En esta opción se especifica el formato de salida del archivo. Las opciones posibles son: Auto, XHTML 1.0 transitional, XHTML 1.0 frameset, XHTML 1.0 strict, XHTML 1.1, XHTML 1.0 transitional, XHTML basic 1.0, XHTML basic 1.1, XHTML mobile profile y XHTML print 1.0
- *Line Length:* Se elige el tamaño en caracteres de cada una de las líneas resultantes de la transformación. Las opciones son: 60, 80, 100, 120.

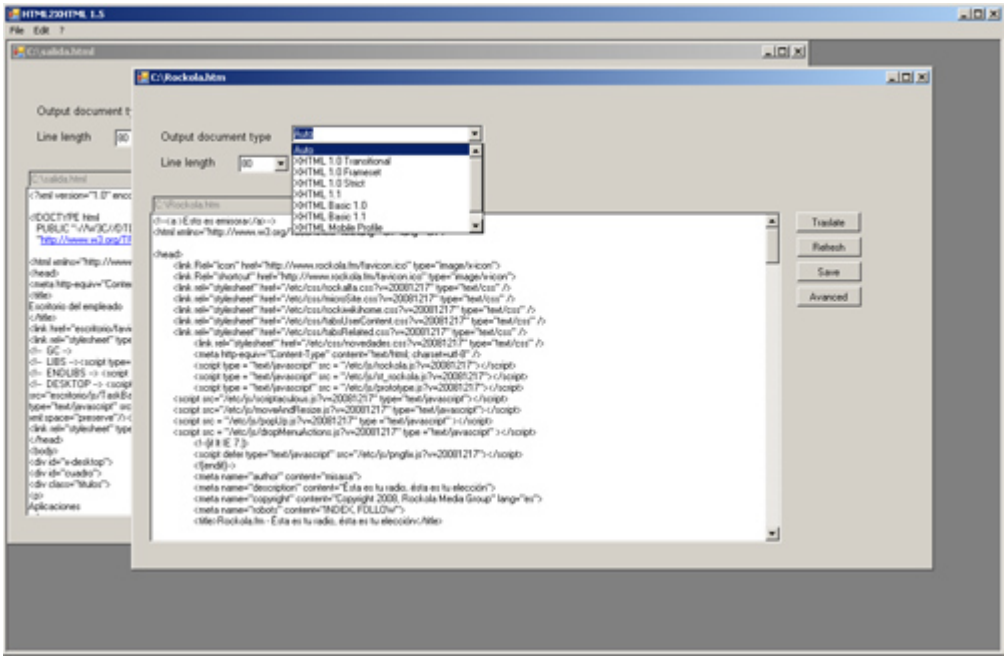



**Figura 23: Distintas opciones de *Output document type.***

A parte de estas opciones, hay otras que no son obligatorias y que están contenidas en la opción *Avanced* situada en último lugar del grupo de botones que están en la parte derecha de la ventana (Figura 25).


**Figura 24: Botones de selección.**

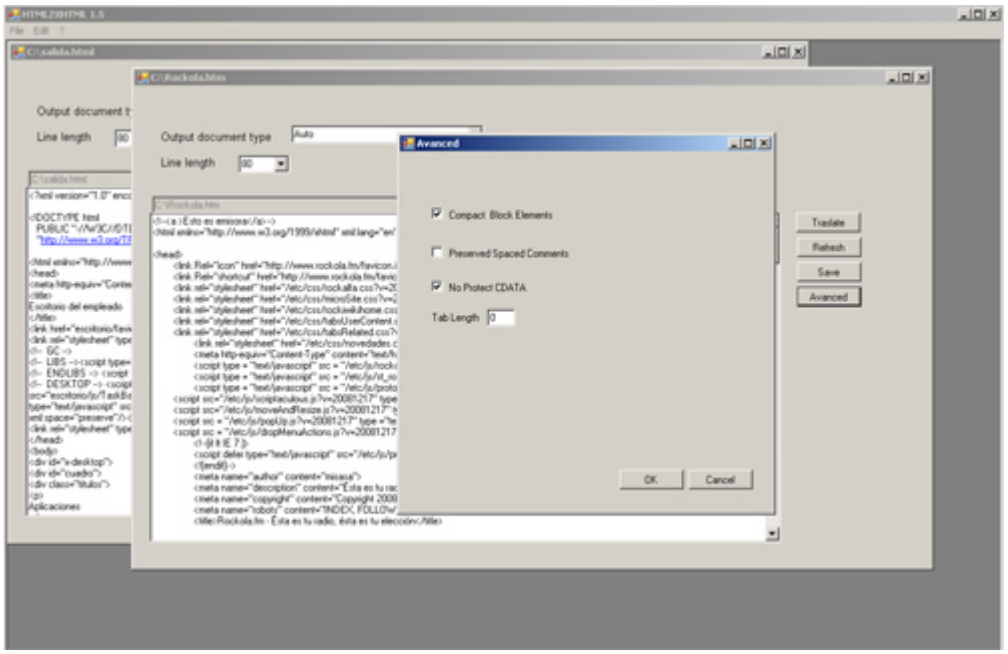



**Figura 25: Ventana Avanced de la aplicación.**

En esta ventana se presentan las distintas opciones disponibles para la transformación del archivo. Estas opciones son las siguientes:

- **Compact Block Elements**: No escribe espacios en blanco o saltos de línea entre el inicio de una etiqueta de bloque de elemento y la etiqueta de inicio de su primer elemento adjunto y entre la etiqueta final de su último elemento adjunto en línea y el final la etiqueta de bloque de elemento.
- **Preserved Spaced Comments**: Utilice esta opción para preservar espacios en blanco, tabuladores y final de las líneas de HTML en los comentarios. El valor por defecto, es reordenar el espacio.
- **No Protect Data**: Incluye las secciones CDATA en "script" y "style" en la especificación XHTML 1.0 (con "<! CDATA [[" y "]]>"). Podría ser incompatible con algunos navegadores.
- **Tab Length**: Tabulador de caracteres. Debe ser un número entre 0 y 16, en otro caso el parámetro es ignorado. El uso de 0 evita el tabulado.

Estas opciones quedan registradas cuando el usuario pulsa el botón *OK*, si no desea guardarlas puede pulsar el botón *CANCEL*.

Si al hacer algún cambio en el código del archivo, hemos cometido uno o varios errores, siempre se podrá volver a cargar el archivo original pulsando el botón REFRESH (Figura 25) con lo que se cargará de nuevo el fichero original.

# TRANSFORMACIÓN DE FICHEROS

La transformación de los ficheros se puede realizar por varios caminos: se puede optar por pulsar el botón *TRASLATE* situado a la derecha del código del archivo, o se puede realizar desde *Menu→File→Traslate…*Además, existe la opción TRASLATE NOW, del menú contextual descrito más arriba.

En cualquier caso, la transformación dará lugar a un nuevo código que será cargado en lugar del antiguo.

Para poder guardar el nuevo código se puede pulsar sobre el botón SAVE que abrirá un cuadro de diálogo que permitirá salvar el fichero en un directorio que deseemos. Así mismo, se puede acceder también desde *Menu→File→Save…*

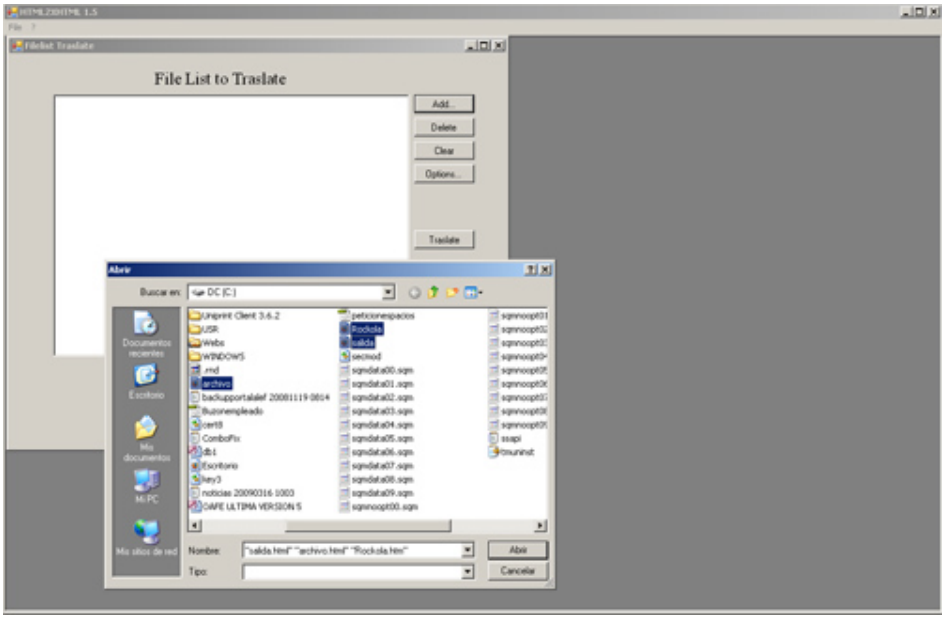



**Figura 26: Ventana de Traducción Múltiple de ficheros con cuadro de diálogo**

Además, la aplicación está diseñada para traducir varios archivos de forma simultánea. Para poder realizar esta función hay que seguir los siguientes pasos:

En la barra de menús, elegir *File→Traslate Multiple Files…* En ese momento aparecerá una nueva ventana como la de la figura 26. En ella, aparece una lista, inicialmente vacía, donde podemos ir colocando los diferentes archivos que queremos traducir.

Mediante el botón *ADD* se pueden ir añadiendo archivos a la lista en formato html (.htm ó .html). Para borrar un elemento, primero se selecciona el archivo a borrar haciendo clic sobre él y a continuación pulsar el botón DELETE para borrarlo de la lista. Se pueden borrar varios elementos simultáneamente seleccionándolos haciendo clic sobre ellos y manteniendo pulsado CRTL a la vez. Si desea borrar toda la lista, basta con pulsar el botón DELETE, y serán eliminados todos los documentos dejándola vacía.

Para poder elegir las distintas opciones de traducción (descritas anteriormente), se debe pulsar el botón *OPTIONS…* para ver la ventana de opciones y poder escoger las opciones necesarias para traducir todos los archivos de la lista (Figura 27).

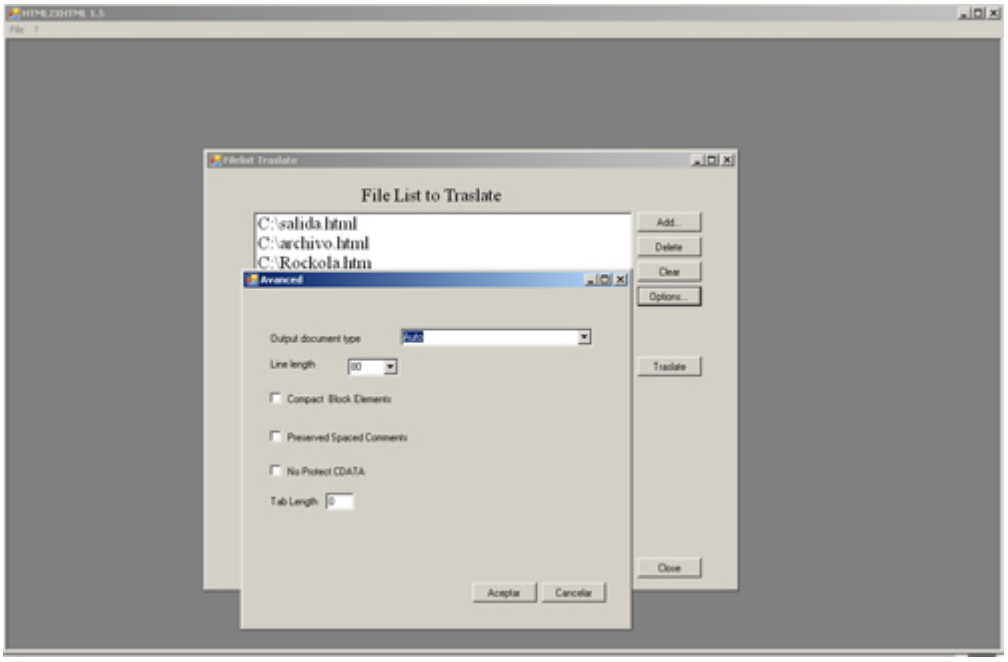



**Figura 27: Ventana de opciones de la lista múltiple.**

Cuando tengamos todos los ficheros que queramos traducir en la lista, solo tenemos que hacer clic en el botón *TRASLATE* para que el programa traduzca

automáticamente los ficheros. Estos archivos se guardaran en el mismo directorio del original, con el mismo nombre y con la extensión .xhtm ó .xhtml según el caso. Al finalizar el proceso, el programa informará de la finalización de la traducción.

# INDICE DE FIGURAS

# GLOSARIO DE TÉRMINOS

**API:** *application programming interface* (interfaz de programación de aplicaciones).

**ASP**: *Active Server Pages* (Páginas de Servidor Activo).

**CERN:** *Conseil Européen pour la Recherche Nucléaire* (Organización Europea para la Investigación Nuclear).

**CIL**: *Common Intermediate Language* (Lenguaje Intermedio Común).

**CLR**: *Common Language Runtime* (Lenguaje común en tiempo de ejecución).

**CLS:** *Common Language Specification* (Lenguaje común de especificación).

**COM** *Component Object Model* (Plataforma de Componentes y Objetos)

**CPU:** *Central Process Unit* (Unida Central de Procesos).

**CSS:** *Cascading Style Sheets* (Hojas de estilo en cascada).

**CTS**: *Common Type System* (Sistema de Tipos Universal).

**DHTML:** D*ynamic HTML* (HTML dinámico).

**DLL:** *dinamic-link library* (Biblioteca de enlace Dinámico).

**ECMA***: European Computer Manufacturers Association*.

**EXE:** *executable* (ejecutable).

**GAC**: *Global Assembly Cache* (Caché de Ensamblados Global).

**GUI:** *graphical user interface* **(**Entornos gráficos de usuario).

**HTML:** *HyperText Markup Language* (Lenguaje de Marcado de Hipertexto*).*

**HTTP**: *Hypertext Transfer Protocol* (protocolo de transferencia de hipertexto).

**IETF:** *Internet Engineering Task Force* (Grupo de Trabajo en Ingeniería de Internet). **JSP:** *Java Server Pages* (Servidor de Páginas Java).

**JVM**: *Java Virtual Machine* (Máquina Virtual de Java).

**MOMA**: *Mono Migration Analyzer*

**NLS:** *oN-Line System* (Sistema On-line).

**ODBC:** *Open DataBase Connectivity* (Conectividad Acceso a Base de Datos).

**SGML:** *Standard Generalized Markup Language (*Lenguaje de Marcado Generalizado).

**XHTML:** e**X**tensible **H**yper**t**ext **M**arkup **L**anguage (lenguaje extensible de marcado de hipertexto).

**XML**: *E**x**tensible **M**arkup **L**anguage* (lenguaje de marcas extensible).

**W3C:** *World Wide Web Consortium.*

**WHATWG:** *Web Hypertext Application Technology Working Group* (Grupo de Trabajo de Aplicaciones Web e hipertexto).

**WWW**: *World Wide Web*.

# BIBLIOGRAFÍA

[1] Bornstein, Niel M y Dumbill, Edd. *Mono: A Developer"s Notebook* Ed. O"REILLY & ASSOCIATES (2004)

[2] CEBALLOS, Fco. Javier. *Aplicaciones .NET multiplataforma - Proyecto Mono* Ed. AlfaOmega (2008)

[3] Charte Ojeda Francisco. *Visual C# .NET* Ed. Anaya Multimedia (2002)

[4] *http://msdn.microsoft.com/es-es/vcsharp/default.aspx Centro de desarrolladores de C# de Microsoft* Microsoft (Visitado 28 junio de 2010).

[5] Nathan, Adam .*NET and COM: The Complete Interoperability Guide* (2002)

[6] Nickell, Seth .*«Why Mono is Currently An Unacceptable Risk»*. Disponible en http://www.gnome.org/~seth/blog/mono (Visitado 2 de Mayo de 2009)

[7] Templeman, Julián y Vitter David, *Visual Studio .NET* Ed. Anaya Multimedia (2002)

[8] Universitat Oberta de Catalunya. *Mono and Gtk# book* Disponible en http://tornatmico.org/libro/LibroMono (Visitado 3 de Mayo de 2009).

[9] Standard ECMA-334 C# Language Specification 4th edition (June 2006) *http://www.ecma-international.org/publications/standards/Ecma-334.htm (*Visitado 15 de abril de 2009*)*

[10] Standard ECMA-335Common Language Infrastructure (CLI) 4th edition (June 2006). *http://www.ecma-international.org/publications/standards/Ecma-335.htm* (Visitado 15 de abril de 2009)

[11] Gnome Foundation, "GNOME: The Free Software Desktop Project", http://www.gnome.org (Visitado 15 de abril de 2009)

[12] IC#SharpCode, *http://www.icsharpcode.net/OpenSource/SD/* (Visitado 15 de abril de 2009)

[13] Microsoft Corp., "Microsoft Developer Network", *http://msdn.microsoft.com* (Visitado 30 de marzo de 2009)

[14] Mono Project, *http://www.mono-project.com/Libraries* (Visitado 15 de abril de 2009)

[15] IETF Network Working Group. *Returning Values from Forms: multipart/form-data*. August 1998. Disponible en http://www.rfc-editor.org/rfc/rfc2388.txt (Visitado 20 de Abril de 2010)

[16] Wikipedia, *http://es.wikipedia.org/wiki/Proyecto_Mono*